Ciencia de datos con R

Gema Fernández-Avilés y José-María Montero

2022-12-22

Índice general

1. Normas CDR-book-v0.2.0 **17**
1.1. Notas importantes . . . 17
1.2. Notas relativas a R . . . 18
1.3. Notas de estilo . . . 19

Prefacio **21**
Los autores . . . 22
¿Por qué este libro? . . . 23
Características . . . 23
El paquete `CDR` . . . 24
¿A quién va dirigido? . . . 24
Agradecimientos . . . 24
Información del software . . . 24

I Ciencia, datos, software... y científicos **25**

2. ¿Es la ciencia de datos una ciencia? **27**
2.1. ¿Qué se entiende por ciencia? . . . 27
2.2. ¿Qué diablos es la ciencia de datos? . . . 28
2.3. Lo científico de la ciencia de datos . . . 30

3. Metodología en ciencia de datos **33**
3.1. Preliminares . . . 33
3.2. Principales metodologías en ciencia de datos . . . 34
3.3. CRISP-DM para ciencia de datos . . . 35

4. Ciencia de datos con R **39**

4.1. Introducción . . . 39

4.2. La sesión de **R** . . . 40

4.3. Instalación de **R** . . . 41

4.4. Trabajar con proyectos de RStudio . . . 41

4.5. Tratamiento de datos con **R** . . . 42

4.6. Organización de datos con el *tidyverse* . . . 47

5. Ética en la ciencia de datos **57**

5.1. ¿Por que la ética en la ciencia de datos? . . . 57

5.2. Los principios éticos . . . 58

5.3. La importancia de los sesgos . . . 59

5.4. ¿Es necesaria la explicabilidad? . . . 61

5.5. Sesgos y explicabilidad en R . . . 62

II Bienvenido a la jungla de datos **65**

6. Gestión de bases de datos relacionales **67**

6.1. Introducción . . . 67

6.2. Concepto de Base de datos . . . 68

6.3. El Lenguaje Estructurado de Consulta (SQL) . . . 69

6.4. Usando bases de datos desde R . . . 72

7. Gesitón de bases de datos NoSQL **83**

7.1. Introducción al Big Data . . . 83

7.2. Las V's del Big Data . . . 84

7.3. Fuentes de Datos en entornos Big Data . . . 85

7.4. Bases de datos Relacionales vs. NoSQL . . . 86

7.5. Bases de datos NoSQL . . . 87

7.6. Ejemplo de integración de una base de datos NoSQL y análisis de datos en R . . . 92

8. Gobierno y gestión de calidad de datos **99**

8.1. Introducción . . . 99

8.2. Concepto de Gobierno de datos . . . 100

8.3. Marcos y metodologías existentes de Gobierno de Datos . . . 103

8.4. Gestión de calidad de datos . . . 105

9. Integración y limpieza de datos **115**

9.1. Introducción . . . 115

9.2. Problemas de calidad de datos . . . 116

9.3. Niveles inadecuados de completitud: Valores *missing* . . . 119

9.4. Mejorando la exactitud y la precisión: eliminación del ruido estadístico . . . 121

9.5. Integración de datos . . . 122

10. Análisis exploratorio de datos **127**

10.1. Introducción . . . 127

10.2. Análisis exploratorio de una característica . . . 131

10.3. Análisis exploratorio de varias características . . . 138

11. Feature selection and engineering **149**

11.1. Introducción . . . 149

11.2. *Feature Selection* (Selección de variables) . . . 151

11.3. Transformaciones de escala y de la distribución de la variable objetivo . . . 157

11.4. *Feature engineering* . . . 159

11.5. Reducción de dimensionalidad . . . 161

11.6. Otras transformaciones . . . 161

III Fundamentos de estadística **171**

12. Probabilidad **173**

12.1. Introducción a la probabilidad . . . 173

12.2. Probabilidad: elementos básicos, definición y teoremas . . . 174

12.3. Variable aleatoria y su distribución: tipos de variables aleatorias . . . 176

12.4. Modelos de distribución de probabilidad . . . 177

12.5. Teorema central del límite 184
12.6. Distribuciones de probabilidad en **R** 184

13. Inferencia estadística **189**
13.1. Introducción a la Inferencia Estadística 189
13.2. Muestreo aleatorio simple 190
13.3. Estimación puntual 192
13.4. Estimación por intervalos 194
13.5. Contrastes de hipótesis 194
13.6. Inferencia estadística paramétrica sobre poblaciones normales 195
13.7. Inferencia sobre poblaciones normales con **R** 197
13.8. Inferencia estadística no paramétrica: contrastes de normalidad 200

14. Muestreo y remuestreo **203**
14.1. Introducción al muestreo 203
14.2. Muestreo aleatorio simple 204
14.3. Muestreo estratificado 207
14.4. Otros tipos de muestreo probabilístico 210
14.5. Técnicas de remuestreo: Bootstrap. 211

IV Modelización estadística **217**

15. Modelización lineal **219**
15.1. Modelización 219
15.2. Procedimiento de modelización 220
15.3. Procedimiento con R: la función `lm` 222
15.4. Casos prácticos 223
15.5. Comentarios finales 231

16. Modelos lineales generalizados **233**
16.1. Motivación 233
16.2. Modelo y sus componentes 234
16.3. Regresión logística 236

16.4. Regresión de Poisson 239
16.5. Casos prácticos 240

17. Modelos aditivos generalizados 251

17.1. Introducción 251
17.2. Splines con penalizaciones 252
17.3. Aspectos metodológicos 252
17.4. La función `gam` del paquete `mgcv` 255
17.5. Casos prácticos 256

18. Modelos mixtos 269

18.1. Conceptos básicos 269
18.2. Formulación del modelo con efectos aleatorios o modelos mixtos 272
18.3. Funciones de `R` para ajustar modelos mixtos 275
18.4. Caso práctico 275

19. Modelos sparse y métodos penalizados de regresión 289

19.1. Introducción 289
19.2. Selección del mejor subconjunto 290
19.3. Selección *Stepwise* 293
19.4. Métodos Shrinkage 296

20. Modelización de series temporales 309

20.1. Conceptos básicos 309
20.2. Modelos ARIMA 310
20.3. Análisis de series temporales con **R** 311

21. Análisis discriminante 325

21.1. Introducción 325
21.2. Tipos de análisis discriminantes: 327
21.3. Procedimiento con **R**: las funciones `lda()`, `qda()` y `mda()` 327

22. Análisis conjunto **335**

22.1. Introducción, conceptos clave y tipos de análisis 335

22.2. Aplicación del Análisis Conjunto (etapas): 337

22.3. Procedimiento con **R**: la función `Conjoint()` 340

23. Análisis de tablas de contingencia **343**

23.1. Introducción 343

23.2. Contraste de independencia en tablas 2×2 347

23.3. Contraste de independencia en tablas $R \times C$ 353

23.4. Medidas de asociación en tablas 2×2 355

23.5. Medidas de asociación en tablas $R \times C$ 358

23.6. Contrastes de independencia en tablas multidimensionales 360

V Machine learning supervisado **363**

24. Árboles de clasificación y regresión **365**

24.1. Introducción 365

24.2. Aprendizaje con árboles de decisión 367

24.3. ¿Cómo se va dividiendo el árbol? 368

24.4. Sobreajuste 370

24.5. ¿Cuánto debe crecer un árbol? 370

24.6. El algoritmo ID3 para la construcción de un árbol de decisión 371

24.7. Procedimiento con R: la funcion `rpart` 372

24.8. Aplicaciones de los árboles de decisión 372

25. Máquinas de vector soporte **383**

25.1. Introducción 383

25.2. Algoritmo SVM para clasificación binaria 385

25.3. ¿Y si tengo más de dos clases? 386

25.4. Truco del Kernel: Tratando con la no linearidad 386

25.5. Procedimiento con R: la funcion `svm` 388

25.6. Aplicación de un modelo SVM Radial con ajuste automático en R 388

26. Clasificador k-vecinos más próximos **395**

26.1. Introducción . . . 395

26.2. Decisiones a tener en cuenta . . . 396

26.3. Procedimiento con R: la funcion knn . . . 398

26.4. Aplicación del modelo KNN en R . . . 398

27. Naive Bayes **403**

27.1. Introducción . . . 403

27.2. Teorema de Bayes . . . 403

27.3. El algoritmo Naive Bayes . . . 404

27.4. Procedimiento con R: la funcion `naive_bayes` . . . 406

27.5. Aplicación del modelo Naive Bayes . . . 407

28. Bagging. Random Forest **411**

28.1. Introducción: Metodos de Ensamble . . . 411

28.2. Bagging . . . 411

28.3. Procedimiento con R: la función `bagging` . . . 412

28.4. Implementando bagging en R . . . 412

28.5. Random Forest . . . 415

28.6. Procedimiento con R: la función `randomForest` . . . 418

28.7. Aplicación del modelo Random Forest en R . . . 419

29. Boosting. XGBoost **427**

29.1. Introducción. Boosting. . . . 427

29.2. Gradient Boosting . . . 428

29.3. Procedimiento con R: la funcion `gbm` . . . 431

29.4. Aplicación del modelo GBM en R . . . 432

29.5. eXtreme Gradient Boosting (XGB) . . . 436

29.6. Procedimiento con R: la funcion `xgboost` . . . 437

29.7. Aplicación del módelo XGBoost en R . . . 437

VI Machine learning no supervisado 441

30. Análisis cluster: clusterización jerárquica 443

30.1. Introducción . . . 443
30.2. Selección de las variables . . . 445
30.3. Elección de la distancia entre elementos . . . 446
30.4. Técnicas de agrupación jerárquicas . . . 450
30.5. Calidad de la agrupación y número de clusters . . . 461

31. Análisis cluster: clusterización no jerárquica 467

31.1. Métodos de reasignación . . . 467
31.2. Métodos basados en la densidad de puntos . . . 472
31.3. Otros métodos . . . 474
31.4. Nota final . . . 475

32. Análisis de componentes principales 477

32.1. Introducción . . . 477
32.2. Obtención de las componentes principales . . . 478
32.3. Estimación de las componentes principales . . . 481
32.4. Número de componentes a retener . . . 482
32.5. Interpretación de las CP . . . 484
32.6. Reproducción de los datos tipificados y de R a partir de las CP . . . 486
32.7. Limitaciones del ACP . . . 487

33. Análisis factorial 489

33.1. Introducción . . . 489
33.2. Elementos teóricos del análisis factorial . . . 490
33.3. El análisis factorial en la práctica . . . 494
33.4. Relaciones y diferencias entre el AF y el ACP . . . 507

34. Escalamiento multidimensional 509

34.1. Introducción . . . 509
34.2. Medición de distancias y similitudes . . . 510
34.3. Modelo escalamiento multidimensional. . . . 512
34.4. Tipos de escalamiento multidimensional . . . 514

35. Análisis de correspondencias **521**

35.1. Introducción . . . 521

35.2. Metodología del análisis de correspondencias . . . 522

35.3. Procedimiento con **R**: la función `ca()` . . . 523

VII Deep learning 531

36. Redes neuronales artificiales **533**

36.1. ¿Qué es el *deep learning*? . . . 533

36.2. Aplicaciones del *deep learning* . . . 535

36.3. Redes Neuronales . . . 538

36.4. Perceptrón o Neurona . . . 538

36.5. Perceptrón Multiclase . . . 540

36.6. Funciones de activación . . . 541

36.7. Perceptrón Multicapa . . . 545

36.8. Instalación de librerías de *deep learning* en **R**: Tensorflow/Keras . . . 548

36.9. Ejemplo de red para clasificación . . . 549

36.10 Ejemplo de red para regresión . . . 556

37. Redes neuronales convolucionales **561**

37.1. Introducción . . . 561

37.2. Convolución . . . 562

37.3. Neuronas convolucionales . . . 564

37.4. Relleno del borde . . . 564

37.5. Capas de agrupación . . . 565

37.6. Desvanecimiento del gradiente . . . 566

37.7. Sobreajuste . . . 568

37.8. Generación de datos de entrenamiento artificiales . . . 569

37.9. Ejemplo para el conjunto de datos `CIFAR10` . . . 571

VIII Ciencia de datos de texto y redes **581**

38. Minería de textos **583**

38.1. Introducción . . . 583

38.2. Conceptos y tareas fundamentales . . . 584

38.3. Análisis de sentimientos . . . 587

38.4. Caso de aplicación . . . 588

39. Análisis de grafos y redes sociales **601**

39.1. Introducción . . . 601

39.2. Teoría de grafos . . . 601

39.3. Elementos de un grafo . . . 602

39.4. El paquete `igraph` . . . 605

39.5. Análisis de influencia en un grafo aplicado a redes sociales . . . 606

39.6. Otras utilidades de grafos . . . 612

IX Ciencia de datos espaciales **617**

40. Trabajando con datos espaciales **619**

40.1. Conceptos clave . . . 619

40.2. Mi primer mapa . . . 627

40.3. ¿Cómo (no) mentir con la visualización? . . . 628

40.4. Mapas espacio-temporales . . . 629

40.5. Mapas interactivos . . . 631

40.6. Estadística para datos espaciales . . . 632

41. Geoestadística **635**

41.1. Introducción . . . 635

41.2. Preliminares . . . 636

41.3. Análisis estructural de la dependencia espacial . . . 638

41.4. Kriging . . . 649

42. Modelos econométricos espaciales **653**

42.1. La dependencia espacial 653

42.2. Medidas de Autocorrelación 657

42.3. Modelos econométricos espaciales de corte transversal 660

43. Procesos de punto **671**

43.1. Spatial point patterns on $\mathbb{R}^2$ 671

43.2. Spatial point patterns on linear networks 682

X Cómunica y colabora **687**

44. Informes reproducibles con R Markdown y Quarto **689**

44.1. Introducción 689

44.2. Documentos Quarto 694

44.3. Otros formatos 699

45. Aplicaciones webs interactivas con Shiny **701**

45.1. Introducción 701

45.2. Partes mínimas de una aplicación `Shiny` y disposición básica 702

45.3. Diseño de una aplicación `Shiny` 703

45.4. Elementos para entrada de datos 706

45.5. Elementos para visualización (salida) 709

45.6. Reactividad 711

45.7. Publicación de la aplicación en la web 713

45.8. Extensiones de `Shiny` 715

46. Git y GitHub en R **717**

46.1. ¿Qué es Git? 717

46.2. ¿Qué es GitHub? 718

46.3. ¿Por qué usar Git y GitHub? 719

46.4. Configuración 719

46.5. Configurar git 720

46.6. Workflow 723

47. Geoprocesamiento en nube **725**

47.1. Sintaxis de Google Earth Engine 725

47.2. Primeros pasos 726

47.3. Cálculo de anomalias 727

XI Casos de estudio en ciencia de datos 731

48. Análisis de una red criminal **733**

48.1. El *dataset Oversize* 733

48.2. Creación del grafo 733

48.3. Visualización del grafo 734

48.4. Importancia de los actores 736

48.5. Identificación de comunidades 737

48.6. Visualización de comunidades 738

49. Optimización de inversiones publicitarias **741**

49.1. Metodologías para optimizar las inversiones publicitarias 741

49.2. Robyn como alternativa open-source en R 743

50. Análisis electoral: de Rstudio a su periódico **751**

50.1. Obtención de los datos 751

50.2. Transformación y primeros gráficos 753

51. Evolución del paro registrado en Castilla-La Mancha **759**

51.1. Planteamiento 759

51.2. Evolución del número total de parados 760

51.3. Evolución de la edad y el sexo en la población parada 761

51.4. Evolución del tiempo de búsqueda de empleo en la población parada 764

51.5. Evolución del paro registrado según sexo, edad y sector de procedencia 765

51.6. Conclusiones 767

52. Valoración del precio de la vivienda 769

52.1. Introducción . . . 769

52.2. Conjunto de datos . . . 770

52.3. Estimación del Modelo . . . 772

53. Segmentación de clientes en retail 781

53.1. Motivación y conceptos clave . . . 781

53.2. Del Modelo RFM tradicional al Modelo RFM extendido, mejoras propuestas . . 781

53.3. Modelo RFM extendido . . . 782

54. Ciencia de datos para la investigación y ensayos clínicos 791

54.1. Justificación . . . 791

54.2. Introducción al uso de datos en investigación clínica y ensayos clínicos . . . 791

54.3. Ejemplo práctico de un estudio observacional . . . 792

54.4. Conclusión . . . 799

55. Lo que nos cuentan los datos sobre el cambio climático 801

55.1. Consideraciones iniciales . . . 802

55.2. Paquetes . . . 802

55.3. Visualización de mapas de pequeños múltiples . . . 802

56. Predicción de demanda eléctrica con deep learning 813

56.1. Introducción . . . 813

56.2. Datos de entrada . . . 813

56.3. Caso de estudio . . . 814

57. Sistemas expertos en el ámbito pediátrico 819

57.1. Introducción . . . 819

57.2. Marco teórico . . . 819

57.3. Sistema experto para el ámbito pediátrico en atención primaria . . . 824

58. El NLP y las tendencias en el mundo de la moda 831

58.1. Introducción . . . 831

58.2. NLP para tendencias de moda en textil . . . 831

59. Detección de fraude de tarjetas de crédito **839**

59.1. Introducción 839

59.2. Modelización del fraude en la compra con tarjetas de crérdito 840

.1. Información de la sesión 845

Capítulo 1

Normas CDR-book-v0.2.0

1.1. Notas importantes

1. La carpeta que acabamos de compartir `book-cdr-v0.2.0-autores` contiene la versión con la que se ha generado tanto el PDF como el gitbook. Por favor, tomad este proyecto como definitivo (por ahora) y modificad vuestros capítulos sobre esta versión que os compartimos.

2. A algunos autores os hemos pedido que, una vez comprobado el código y generado el capítulo, **formateéis las chunks con `eval=FALSE` (para que no ejecute) y luego incluyáis el resultado, plot, etc**. Otras veces las habéis puesto vosotros directamente. **Esto se agradece y siempre que podáis, hacedlo, please**, pues como supondréis compilar todo y con tantos paquetes distintos ha dado bastantes errores.

3. No llaméis directamente al paquete `tidyverse`, pues tenemos/tengo conflictos, empezó todo con `filter()` y luego se añadieron más funciones. Por ahora está solucionado.

4. Revisad bien el PDF y el gitbook (https://cdr-book.github.io/prefacio.html) y comprobad que está todo y bien. Importante, github distingue mayúsculas y minúsculas, luego los nombres las figuras deben coincidir. Es decir, si la figura se llama "portada-cdr.jpg", y ponemos "portada-cdr.JPG", aunque se genere el HTML y el PDF, como github.io distingue mayúsculas y minúsculas no generaría la imagen.

5. Ejecutad desde el PDF/HTML el código y comprobar que se generan perfectamente los resultados esperados.

6. Ya están etiquetados todos los capítulos, así que ya se pueden hacer referencias cruzadas, cosa que conviene para dar continuidad al libro.

7. Según vayáis revisando, etiquetad también todas las chunks, con un indicativo del capítulo para que no haya duplicidades. Al compilar todo había errores y al no estar etiquetas las chunks ha sido muy difícil ver donde estaba el error.

8. Es importante hacer referencias cruzadas en los capítulos/secciones y eliminar duplicidades. Todos nos hemos pasado de la estimación inicial de páginas. El formato del PDF ahora mismo es 19×25 en vez de 17×24 que era el original. Puede ser una opción si McGraw lo acepta (lo han sugerido ellos pero ahora lo tienen que aprobar). La extensión es de 853 págs. en vez de 700 págs...

9. Comprobad que los datos están en el paquete del libro, `CDR`, con su descripción correspondiente y que en el capítulo se llaman directamente desde la librería, no desde la carpeta `data/file.csv`

```
library("remotes")
install_github("cdr-book/CDR")
library("CDR")
data(package = "CDR")
```

10. Las referencias bibliográficas están revisadas (pues había duplicidades) e incluidas en el archivo `index.Rmd`

11. Generad las palabras clave con el comando `\index{}` de latex.

1.2. Notas relativas a R

- Para el estilo del código, estamos siguiendo `styler`. https://www.tidyverse.org/blog/2017/12/styler-1.0.0/
- Un paquete en el texto irá con la comilla invertida: `CDR` y citado la primera vez..
- Una función en el texto irá con la comilla invertida y con paréntesis: `data()`.
- Las referencias a **R** en el texto irán en negrita: **R**
- Todos los objetos, variables, argumentos, etc. de R con la comilla invertida. Por ejemplo:

...incluye la función `preProcess(x, method)`, que se puede integrar en el entrenamiento (función `train()`) para estimar...

A continuación, se muestra su uso con un ejemplo del dataset `Madrid_Sale`, incluido en el paquete de **R** `Idealista18`.

- **Nombres de las variables y objetos** que se crean:
  - unidos por guión bajo: `reg_log` en vez de `reg.log`.
  - informativos y lo más cortos posibles: `med_pib_filtro` en vez de `media_pib_filtro`
- Cuando se cargan los datos de un paquete, especificarlo en la chunk. Por ejemplo:

```
library("gapmaminder")
data("gapmaminder")
```

1.3. Notas de estilo

1.3.1. Títulos

- Solo la primera letra en mayúscula. Ejemplos:
  - **Redes neuronales artificiales**
  - **Gobierno y gestión de calidad de datos**
- Lo suficientemente informativos y cortos para que quepan en el encabezado.
- Mejor en español, en la medida de lo posible y entre paréntesis en inglés. Ejemplo:
  - Selección de variables (*Feature Selection*) y no al revés.

1.3.2. Figuras

- Tienen que verse bien la leyenda o el texto que contenga y ser de calidad.
- Deben tener su *caption* informativa, por ejemplo, fig.cap='Comparativa de metodologías de medición.'} y estar etiquetadas para luego referenciarlas.
- Referenciarlas en el texto con "Fig.". Ejemplo: La Fig. XX muestra ...
- Dibujarlas a color (el libro es a color).
- Se puede utilizar la librería `patchwork` (por ejemplo) para formatear los cuadros (ver https://patchwork.data-imaginist.com/)
- Incluid las figuras estáticas, contenidas en la carpeta **img**, mejor con esta opción

```
knitr::include_graphics("img/Comparativa_modelos.PNG")
```

1.3.3. Tablas

- Las tablas hay que generarlas en el texto con Rmardown. Si la tabla es muy grande igual es conveniente con Latex. Me ha pasado un autor que ha hecho bastantes tablas en sus capítulos es siguiente enlace, muy útil y que funciona muy bien. https://www.tablesgenerator.com/
- Todas tienen que tener su título y estar etiquetadas para luego referenciarlas.
- Tienen que estar citadas en el texto. La Tabla xx muestra...

1.3.4. Otros

- Escribid siempre en impersonal y repasad tildes, etc.
- Al hacer referencia a capítulos y secciones escribir: Cap. XX, Sec. XX.
- "Formatead" el resultado del PDF controlando figuras y texto y sabiendo que todos los capítulos empiezan en página impar.
- En el cuadro resumen, color amarillo, poned **Resumen** (sólo la primera en mayúscula y en negrita, más un intro) y en el cuadro de notas, **Nota** (sólo la primera en mayúscula y en negrita, más un intro).

Resumen
Aquí el texto

Nota
Aquí el texto

- Palabras inglesas en cursiva. Ejemplo: valores *missing*
- Evitad "como ya se ha mencionado," y decid, "como se ha mencionado en el Cap. XX"
- Minúscula depués de : en español. Por ejemplo,
  - `formula`: refleja el...
- Conceptos importantes en **negrita**.
- Recordad que se puden utilizar los entornos disponibles para ejemplos, ejercicios, etc...

Ejemplo 1.1. 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**

Prefacio

Nota: Este libro está publicado por McGraw Hill. Las copias físicas están disponibles en McGraw Hill. La versión online de este libro se puede leer de forma gratuita en https://cdr-book.github.io/ y tiene la **licencia FALTA**. Si tiene algún comentario, no dude en contactar con los editores/autores. ¡Gracias!

Los autores

TO DO

Aquí listado de nombre y afiliaciones por orden alfabético?

- `Pedro Albarracín García.` FDS a DXC Technology Company.
- `Jose-Luis Alfaro Navarro.` Universidad de Castilla-La Mancha.
- `Eusebio Angulo Sánchez-Herrera.` Universidad de Castilla-La Mancha
- `Borja Andrino.` El País.
- `Itzcóatl Bueno García.` Instituto Nacional de Estadística
- `Ismael Caballero Muñoz-Reja`. Universidad de Catilla-La Mancha. DQ-Team
- `Miguel Camacho-Collados.` Policía Nacional
- `Ramón Carrasco González.` Universidad Complutense de Madrid
- `Víctor Casero Alonso.` Universidad de Catilla-La Mancha
- `Coro Chasco Yrigoyen.` Universidad Autónoma de Madrid
- `Ángela Celis Sánchez.` Universidad de Castilla-La Mancha
- `Luz Congosto Martínez.` Univesidad Carlos III de Madrid
- `María Durbán Reguera.` Universidad de Carlos III de Madrid
- `José Luis Espinosa Aranda.` Ubotica
- `Gema Fernández-Avilés Calderón.` Universidad de Castilla-La Mancha
- `Jaime Fierro Martín.` Analitycae
- `M. Carmen García Centeno.` Universidad San Pablo-CEU
- `José J. Galán Hernández.` Universidad Complutense de Madrid
- `Rocío González Martínez.` Analitycae
- `Aurora González Vidal.` Universidad de Murcia
- `Isidro Hidalgo Arellano.` Junta de Comunidades de Castilla-La Mancha
- `Ángel Jiménez Rojas.` Junta de Comunidades de Castilla-La Mancha

`Jesús Lago Millas.` Scout Analyst

- `Federico Liberatore.` Cardiff University

- `Emilio López Cano`. Universidad Rey Juan Carlos
- `Bilal Laouah`. People Analytics
- `María Lozano Zahonero`. Università di Roma Tor Vergata
- `Ambrosio Nguema`. Inditex
- `Jorge Mateu Mahiques`. Universidad Jaime I de Castellón
- `José-María Montero Lorenzo`. Universidad de Castilla-La Mancha
- `Leticia Meseguer Santamaría`. Universidad de Catilla-La Mancha
- `Román Mínguez Salido`. Universidad de Castilla-La Mancha
- `Mehdi Moradi`. Umea University
- `José Ángel Olivas Varela`. Universidad de Castilla-La Mancha
- `Arturo Peralta Martín-Palomino`. Universidad Internacional de la Rioja
- `Ricardo Pérez del Castillo`. Universidad de Catilla-La Mancha
- `Lara Quijano-Sánchez`. Universidad Autónoma de Madrid
- `Dominic Royé`. Universidad de Santiago de Compostela
- `Carlos Real Ugena`. Deloitte
- `José M. Sanz Candales`. Red Eléctrica Española
- `Cristina Sánchez Figueroa`. Universidad Nacional a Distancia
- `Jorge Velasco López`. Instituto Nacional de Estadística
- `Manuel Vargas Vargas`. Universidad de Catilla-La Mancha
- `Noelia Vállez Enano`. Universidad de Catilla-La Mancha
- `Andrés Vallone`. Universidad Católica de Chile

¿Por qué este libro?

TO DO

Características

TO DO

Partes, capítulos, casos de uso, R, académicos y profesionales, ...

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()`

```
# este comando solo necesita ser ejecutado una vez
remotes::install_github("cdr-book/CDR")
```

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

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

¿A quién va dirigido?

TO DO

Agradecimientos

TO DO

Información del software

TO DO

Parte I

Ciencia, datos, software... y científicos

Capítulo 2

¿Es la ciencia de datos una ciencia?

Gema Fernández-Avilés y *Bilal Loual*

2.1. ¿Qué se entiende por ciencia?

No es posible determinar si la ciencia de datos es una ciencia sin previamente consensuar la definición de ciencia. La palabra **ciencia** *'Scientia'* proviene del verbo **saber** *'scire'*. Por tanto, en su origen, el saber y la ciencia se consideraban exactamente lo mismo. En este sentido, Bunge (2004) distingue dos categorías fundamentales de conocimiento: el conocimiento vulgar y el científico. El primero se adquiere de manera cotidiana a partir de percepciones y sensaciones individuales, apoyándose en la evidencia y el sentido común. Este tipo de conocimiento constituye la base sobre la que se sustenta el conocimiento científico y éste se enriquece al verificar, con la ayuda de un método, la validez de las observaciones realizadas. Se adquiere, por tanto, de forma consciente, deliberada y metódica, pudiendo ser sometido a prueba y, llegado el caso, podría ser superado, Montero (1997).

Una definición generalmente aceptada de ciencia es al propuesta por Blaug (1980): "*ciencia es el cuerpo de proposiciones sintéticas acerca del mundo real que es susceptible, al menos en principio, de falsaciones por medio de la observación empírica, ya que excluye la posibilidad de que ciertos acontecimientos se produzca. Así pues, la ciencia se caracteriza por su método de formulación de proposiciones contrastables, y no por su contenido, ni por su pretensión de certeza en el conocimiento; si alguna certeza proporciona la ciencia, ésta será más bien la certeza de la ignorancia.*"

Si bien esta definición está sesgada hacia el enfoque popperiano, introduce la palabra clave de la discusión planteada: el **método**, y más concretamente, el **método científico**.

La palabra método procede del latín *'methŏdus'*, y éste del griego $\mu\varepsilon\theta o\delta 0\varsigma$, que quiere decir **camino o sendero que conduce a un fin**. Es, pues, un procedimiento para conseguir algo, y

como el fin que busca la ciencia es la verdad, el método científico será el camino mediante el cual las ciencias encuentren sus respectivas verdades. El método científico es, por tanto, el conjunto de procedimientos por los cuales se adquieren conocimientos rigurosos, ciertos y seguros acerca de un objeto. El método científico ha de ser recorrido siguiendo cuatro fases (Cancelo, 1997): (i) Un previo inventario de los fenómenos o de los hechos significativos no rutinarios; (ii) el planteamiento de un tema problemático que hace necesaria una explicación; (iii) la ideación de conjeturas tendentes a darla y (iv) el tratamiento de las diversas hipótesis hasta que sólo una se mantenga.

Por tanto, aunque no hay acuerdo a la hora de dar una definición exacta de ciencia, sí hay un mayor acuerdo a la hora de aceptar el método científico como elemento que define la frontera de la ciencia: "*El método científico y la finalidad a la cual se aplica (conocimiento objetivo del mundo) constituyen la entera diferencia que existe entre la ciencia y la no-ciencia [...]. El método científico es un rasgo característico de la ciencia, tanto de la pura como de la aplicada: donde no hay método científico no hay ciencia*", Bunge (2004). "*Una ciencia es una disciplina que utiliza el método científico con la finalidad de hallar estructuras generales (leyes).*" Bunge (2004).

Este desplazamiento de la idea central alrededor de la cual se articula el concepto de ciencia es el que permite pasar a discutir acerca de si cabe hablar o no de Ciencias de datos como ciencia.

2.2. ¿Qué diablos es la ciencia de datos?

La ciencia de datos es una disciplina emergente donde, a diferencia de otros saberes como, por ejemplo, las ciencias matemáticas, el *corpus* o la acumulación de conocimiento se ha generado en un lapso de tiempo relativamente corto y de una forma muy intensiva, y no a lo largo de siglos de historia. Su inicio data de la década de los 70, aunque en 1962 el estadítsico John W. Tukey precedió al término "ciencia de Datos" en su artículo "*The Future of Data Analysis*" al explicar una evolución de la estadística matemática. En éste, definió por primera vez el análisis de datos como: "*Procedimientos para analizar datos, técnicas para interpretar los resultados de dichos procedimientos, formas de planificar la recopilación de datos para hacer su análisis más fácil, más preciso o acertado, y toda la maquinaria y los resultados de las estadísticas matemáticas que se aplican al análisis de datos*", (Tukey, 1962). A partir de este momento, toda una serie de acontecimientos fueron consolidando el termino de ciencia de datos como una nueva disciplina. Una breve descripción de los acontecimientos se muestran en la Fig. (2.1).

La ciencia de datos implica la limpieza, la agregación y la manipulación de datos, adquiridos de la web, de teléfonos inteligentes, de clientes, de pacientes, de sensores, de encuestas,... entre otras fuentes, para realizar la modelización y el análisis de datos avanzado que ayuden a identificar patrones, tendencias, comportamientos y, por tanto, a la toma de decisiones. El volumen acelerado de fuentes de datos y, posteriormente, de datos, ha hecho que la ciencia de datos sea uno de los campos de más rápido crecimiento en todas las industrias. Como resultado, no sorprende que el científico de datos surgiera como una nueva profesión que ayudaría a comprender y a analizar los volúmenes masivos de datos que se acumulaban en ese momento y que fuese calificado como el "trabajo más sexy del siglo XXI" según Davenport and Patil (2012).

Figura 2.1: Linea del tiempo de la ciencia de datos.

La ciencia de datos es, por tanto, una disciplina relativamente nueva que combina la Estadística, la Matemática, la Informática y la Programación, para obtener valor de los datos. Se utiliza en una amplia variedad de campos, como la Astronomía, la Medicina, la Economía, el Marketing, las Finanzas, la Biología, la Industria, etc. Esta naturaleza transdisciplinaria de la ciencia de datos añade cierta complejidad a su caracterización, pues siendo una única disciplina, subsume en su ejercicio otras disciplinas como la ciencia Matemática y Estadística y la ciencia de la Computación, que a su vez son aplicadas a un amplio rango de dominios de manera integral. La ciencia de datos se sirve de los métodos formales de la Matemática y de las aplicaciones prácticas e ingenieriles de las Ciencias de la Computación para la generación de conocimiento y para la resolución de problemas prácticos en múltiples campos. Esta ubicuidad hace que la ciencia de datos esté situada transversalmente entre los saberes de primer orden. En otras palabras, la ciencia de datos va adoptando los paradigmas, modelos, teorías o constructos propios del campo sustantivo en el que se ejerce, de forma que para resolver alguna problemática sobre personas, puede recurrirse al *corpus* relativo de la Psicología o de la Sociología y para profundizar sobre alguna condición de salud, podrá hacerse lo propio con la Medicina o la Biología, por mencionar algunos ejemplos.

2.3. Lo científico de la ciencia de datos

En la Sección 2.1 se manifestó que un aspecto fundamental de la ciencia es que utiliza el método científico con la finalidad de hallar estructuras generales (principios y leyes) con capacidad predictiva y comprobable. Es por ello que el marco general de la **metodología científica** ha sido bien fundamentado a lo largo de las últimas décadas a través de las contribuciones de diferentes teóricos de la ciencia (Díez and Moulines (2008), Chalmers et al. (2000), Bunge (2004)). Por otra parte, la ciencia se clasifica según el objeto de estudio en (Bunge, 2018): empíricas y formales[1]. Dado que la ciencia de datos se aplica a diferentes campos, puede tomar las características de las ciencias empíricas y de las formales.

Si se analiza el conjunto de saberes científicos se aprecia que tienen en común una serie de características (Bunge, 2018). La pregunta fundamental en este punto, por tanto, es: ¿Comparte la ciencia de datos estas características? De ser satisfechas conferirían a la ciencia de datos el estatuto de ciencia que comparten otros saberes científicos:

(I) La actividad científica es **metódica**. Es decir, se caracteriza por proceder de manera ordenada y planificada. Esta estructuración le otorga solidez y consistencia. En ciencia de datos también se actúa de manera metódica, a través de diferentes metodologías, como lo son *Knowledge Discovery in Databases* (KDD), *Sample, Explore, Modify, Model, Assess* (SEMMA) y *CRoss-Industry Standard Process for Data Mining* (CRISP-DM), tal y como se expone en el capítulo @ref{metodología}.

(II) El conocimiento científico **se fundamenta en hechos**. Por lo común, los científicos disponen de diferentes instrumentos para observar y registrar la realidad sobre la que

[1] Las ciencias empíricas emplean una aproximación hipotético-inductivo-deductiva y amplían el conocimiento. Las ciencias formales se fundamentan en la deducción y explicitan el saber, pero no lo amplían. Ejemplo de ciencia empírica lo es la Biología, mientras que la Lógica lo es de la ciencia formal.

conjeturan. Esta labor también es realizada por los científicos de datos, quienes cuentan con un número amplio de instrumentos y metodologías para la recolección de datos. Tal es el caso de los cuestionarios, escalas psicométricas o datos transaccionales producidos por diferentes tecnologías.

(III) El saber científico es **falsable**. Ésto implica que las afirmaciones científicas pueden ser contrastadas, precisamente, a través de los hechos. En ciencia de datos, esto también sucede, ya que los resultados a los que se llega no están ligados a la subjetividad del analista, sino a la objetividad de los datos.

(IV) La ciencia es una actividad que **trasciende los hechos**. Es decir, la ciencia parte de evidencias empíricas que tienden a ser superadas, puesto que la explotación de las mismas suele generar nuevas evidencias que a su vez pueden contribuir a crear nuevos marcos teóricos explicativos o a ampliar los existentes. La ciencia de datos puede ejercerse en el mismo sentido. Por ejemplo, la construcción de un recomendador como Netflix, parte de ciertos datos, pero su uso genera nuevos datos de comportamiento que pueden ser empleados para optimizar su sustrato algorítmico.

(V) La investigación científica se caracteriza también por ser una **actividad analítica**. Es decir, tiende a descomponer los problemas en sus partes constitutivas. Obsérvese que la consecuencia de ello es que no se pueda hablar de una ciencia general, sino de especializaciones. Naturalmente, la especialización también existe en esta disciplina, por eso, cuando la ciencia de datos se aplica intensivamente en Recursos Humanos, por ejemplo, es posible hablar de *Human Resource Analytics*.

(VI) La ciencia es **comunicable** y para ello se sirve de sistemas representacionales lógico-formales. Este atributo también se aprecia en la ciencia de datos, puesto que los hallazgos tienden a ser compartidos a través de diferentes estrategias, entre ellas, la visualización de datos.

La ciencia, sin embargo, no sólo puede describirse en términos de características, sino también **funcionalmente** (Hempel (2005)). De hecho, las características anteriormente citadas son las que posibilitan las funciones **descriptiva**, **explicativa** y **predictiva**.

- La primera, la **descriptiva**, permite recabar información sobre el suceso que se analiza para tratar de conocerlo en mayor profundidad y detalle. En ciencia de datos, usualmente, una de las primeras tareas consiste en describir el conjunto de datos para conocer en detalle sus características, es decir, el número de variables, el número de observaciones, los valores nulos, etc., (esta parte se conoce como "Entendimiento de los datos" en la metodología CRIPS-DM, ver Sección 3.3).

- La segunda, la **explicativa**, determina cómo se relacionan los fenómenos que se observan. Por ejemplo, cuando un científico de datos emplea un modelo de regresión lineal, lo que hace es establecer una relación explicativa entre la variable dependiente y las independientes a través de los coeficientes, (esta parte se conoce como "Validación" en la metodología CRIPS-DM, ver Sección 3.3).

- La tercera, la **predictiva**, permite anticipar ciertos eventos en el tiempo. Tal es el caso de los científicos de datos que ejercen su labor en el ámbito comercial y emplean, por ejemplo, el análisis de series temporales para pronosticar las ventas futuras y poder realizar una planificación del aprovisionamiento de *stocks* con mayor eficiencia, (esta parte está incluida en la fase de "Modelado" en la metodología CRIPS-DM, ver Sección 3.3).

De todo lo expuesto hasta aquí, se puede sostener, sin lugar a dudas, que la ciencia de datos emplea el método científico y comparte las principales funciones de la ciencia. Ahora bien, la ciencia de datos no puede entenderse plenamente sin presuponer las otras disciplinas en las que se aplica. Por tanto, uno de los interrogantes que deberán resolver los futuros profesionales es si la ciencia de datos es un saber de primer orden que lidia directamente con la realidad, como la Física o la Química o si, por el contrario, es un saber de segundo orden, es decir, una suerte de disciplina que se sirve de otros saberes para desplegarlos y actualizarlos.

RESUMEN
Para determinar si la ciencia de datos es una ciencia en primer lugar se debe consensuar la definición de ciencia, que va íntimamente ligada a la definición de método científico. Por otra parte, las ciencias tienen en común una serie de características, que deben ser satisfechas por la ciencia de datos para adquirir el estatus de ciencia.
Del análisis anterior, se concluye que la ciencia de datos emplea el método científico y comparte las principales funciones de la ciencia, dejando la respuesta abierta al investigador.

Capítulo 3

Metodología en ciencia de datos

Gema Fernández-Avilés y Ramón A. Carrasco

3.1. Preliminares

En el Capítulo 2 se ha puesto de manifiesto que el método científico es el elemento que define la frontera de la ciencia. Bunge (2018) al hablar del método científico, lo define como: "*Un procedimiento para tratar un conjunto de problemas. Cada clase de problemas requiere un conjunto de métodos o técnicas especiales. Los problemas del conocimiento, a diferencia de los del lenguaje o los de la acción, requieren la invención o la aplicación de procedimientos especiales adecuados para los varios estadios del tratamiento de los problemas...*". De acuerdo con su **concepción del método**, Bunge (2018) destaca ocho operaciones necesarias en la aplicación de este:

(I) Enunciar preguntas bien formuladas y verosímilmente fecundas.

(II) Arbitrar conjeturas, fundadas y contrastables con la experiencia, para contestar las preguntas.

(III) Derivar consecuencias lógicas de las conjeturas.

(IV) Arbitrar técnicas para someter las conjeturas a contrastación.

(V) Someter a contrastación esas técnicas para comprobar su relevancia y la fe que merecen.

(VI) Llevar a cabo la contrastación e interpretar sus resultados.

(VII) Estimar la pretensión de verdad de las conjeturas y la fidelidad de las técnicas.

(VIII) Determinar los dominios en los cuales valen las conjeturas y las técnicas, y formular los nuevos problemas originados por la investigación.

A su vez, sugiere una serie de reglas para la ejecución ordenada de las operaciones anteriores:

(I) Formular el problema con precisión y, al principio, específicamente.

(II) Proponer conjeturas bien definidas y fundadas de algún modo, y no suposiciones que no comprometan en concreto ni tampoco con ocurrencias sin fundamento visible.

(III) Someter las hipótesis a contrastación dura, no laxa.

(IV) No declarar verdadera una hipótesis satisfactoriamente confirmada; considerarla, en el mejor de los casos, como parcialmente verdadera.

(V) Preguntarse por qué la respuesta es como es y no de otra manera.

Sin embargo, de acuerdo con Montero (1997), estas reglas no son definitivas ni infalibles y necesitan de ulterior perfeccionamiento, que se llevará a cabo a lo largo de la investigación científica. Además, las reglas del método científico no son autosuficientes, necesitan apoyarse en la inteligencia y creatividad humanas.

En resumen, es el tratamiento sistemático de los problemas de la forma descrita, y no la certeza de los resultados obtenidos o la utilización de las técnicas muy concretas y específicas, el que garantiza el carácter científico de las conclusiones, Cancelo (1997). Y, la ciencia de los datos, como no podía ser de otra forma, proporciona una serie de metodologías que guían el trabajo de los científicos de datos. Las principales metodologías se presentan a continuación.

3.2. Principales metodologías en ciencia de datos

En un proyecto de ciencia de datos es muy importante la metodología, pues proporciona al científico de datos una estrategia y un marco sobre el que trabajar. Desde finales del siglo XX se han ido proponiendo diversas metodologías enfocadas a la resolución de problemas concretos de negocio mediante el uso de los datos que hoy podrían englobarse en el paraguas común de la ciencia de datos. Destacan tres metodologías: (i) **KDD**, propuesta por Fayyad et al. (1996) e inspirada en un trabajo previo de Brachman and Anand (1994), fue la primera metodología aceptada por la comunidad científica. Se trata del primer intento serio de sistematizar el proceso conocido hoy día como ciencia de Datos y en aquellos tiempos como conocimiento basado en bases de datos, pues se centraba en la minería de datos. (ii) **SEMMA**, cuyas letras coinciden con la inicial de las etapas de las que consta en inglés (Sample, Explore, Modify, Model, and Assess) fue desarrollada y mantenida por el Instituto SAS en 2012. Se define como el proceso de selección, exploración y modelización de grandes bases de datos para descubrir patrones de negocio desconocidos. Y (iii) **CRISP-DM**, planteada inicialmente en 1996 y publicada formalmente en Chapman et al. (2000a), ha sido mantenida durante varios años por la compañía SPSS, posteriormente adquirida por IBM que se ha encargado de mantener y refinar la metodología hasta tiempo actuales. Esta metodología define una secuencia flexible de seis fases que permiten la construcción e implementación de un modelo de minería de datos para ser utilizado en un entorno real, ayudando a apoyar decisiones de negocios. Esta metodología, la más utilizada en la actualidad (Shafique and Qaiser (2014); Azevedo and Santos (2008), entre otros), se describe en la siguiente sección.

3.3. CRISP-DM para ciencia de datos

La metodología CRISP-DM está basada en seis etapas, representadas en la Fig. **??**fig:crisp-dm) que no han variado desde su formulación en 2000 y en una serie de funciones que se han sido refinando en el tiempo en el tiempo CRISP-DM (2021). Las etapas del CRISP-DM de manera esquemática son:

1. **Entendimiento del negocio**. Fundamental para el éxito del mismo. Consta de cuatro fases:

   - *Determinación de los objetivos de negocio*, consensuados previamente con la organización. Es importante fijar los correspondientes KPIs (Key Performance Indicators) que permitan medir fidedignamente dichos objetivos.
   - *Evaluación de la situación actual*, inventariar las fuentes de datos que estarán disponibles, los recursos materiales y humanos que se dispondrán, los factores de riesgo y el plan de contingencia para los mismos.
   - *Determinación de los objetivos del proyecto*, que debe traducirse al correspondiente rendimiento de los modelos (por ejemplo, cual debe de ser su nivel de precisión).
   - *Plan del proyecto*, con sus procesos a realizar y recursos asignados.

2. **Comprensión de los datos**. Es necesario el completo entendimiento de la materia prima (los datos). Está formada por cuatro fases que giran en torno a los datos:

   - *Recopilación*, tanto de datos internos como externos a la organización si son necesarios, incluso se generan datos adicionales, el etiquetado de casos no clasificados con anterioridad.
   - *Descripción*, especificando aspectos como la cantidad de datos disponibles, anticipando posibles problemas de rendimiento, tipología de las variables (numéricas, categóricas, booleanas, etc), codificación de las mismas (especialmente para las categóricas), etc.
   - *Exploración*, a tavés del AED (Análisis Exploratorio de Datos). Esta tarea ayuda a formular hipótesis sobre los datos y dirige las posteriores etapas de preparación y modelado.
   - *Verificación de la calidad*, buscando problemas como valores perdidos, errores de datos (por ejemplo, tipográficos), errores de las mediciones (datos que son correctos en su introducción pero sus unidades de medida no lo son), incoherencias en la codificación (especialmente para variables categóricas).

3. **Preparación de los datos**. Esta etapa suele ser la más costosa en tiempo y esfuerzo del proyecto (frecuentemente más de 60 %). Consta de cinco fases:

   - *Selección*, se toman decisiones sobre los casos o filas a seleccionar y sobre los atributos (variables) o columnas a incluir.

- *Limpieza*, si en la subfase de verificación de la calidad de los datos se han detectado problemas ahora hay que subsanarlos. Los valores perdidos se pueden excluir o interpolar; los errores de datos, se pueden corregir con algún esquema lógico o manualmente; incoherencias en la codificación, se podría recodificar sustituyendo los datos originales.
- *Construcción*, a partir de los ya disponibles, nuevos atributos (variables) o columnas y nuevas filas o registros.
- *Integración*, necesaria para para construir un concepto de negocio unificado (por ejemplo, el concepto de cliente) si se han usado diversas fuentes (tiquet de compra y registos de cliente). La fusión de columnas con algunas claves en común (*join*), adición de filas con las columnas en común (union), agrupación, etc, se utilizan frecuentemente.
- *Formateo*, orientada a las necesidades de los posteriores modelos a usar. La conversión de variables categóricas a numéricas (usando técnicas de *one hot encoding*), la normalización (usando normalizaciones min-max o z-score), etc, son tareas comunes en esta etapa.

(IV) **Modelado**, se trata de que los modelos ingieran dichos datos y aprendan de ellos a resolver el problema de negocio planteado. Las fases de las que consta esta etapa son:

- *Selección de técnicas de modelado*, por ejemplo, si se va a usar Machine Learning supervisado o no supervisado, y tipo de algoritmos a usar más especificadamente dentro de cada una de esta técnicas. Por supuesto, se tienen en cuenta los requisitos fijado en la primera fase, la cantidad y tipo de datos de los que disponemos, los requisitos específicos de cada modelo, etc.
- *Generación de un diseño de comprobación*, a través de medidas y criterios de bondad del modelo (curva ROC, AIC, R^2, Matriz de confusión, etc).
- *Generación de los modelos*, se suelen generar diversos modelos entrenándolos oportunamente para seleccionar posteriormente el más adecuado.
- *Evaluación del modelo*, en base a los modelos generados y al plan de pruebas especificado.

(V) **Evaluación**. Se debe comprobar que el modelo final generado cumple las expectativas de negocio especificadas en la primera fase. Hay que hacer hincapié en este aspecto ya que suele confundir en la práctica esta fase de evaluación con la subfase de la anterior etapa de evaluación del modelo. Ahora la validación es desde el punto de vista de negocio. Así, por ejemplo, cabe plantearse si con el modelo elegido se pueden alcanzar las metas de negocio especificadas y medidas con los correspondientes KPIs. Tras esta evaluación de los resultados del modelo se abre un proceso de revisión que nos permita valorar si finalmente el modelo cumple las expectativas o tenemos que volver a etapas anteriores.

(VI) **Implementación**. El conocimiento obtenido con el modelado es puesto en valor en esta fase de cara a satisfacer los objetivos de negocio planteados en el proyecto. Este despliegue depende mucho del tipo de proyecto que se esté realizando aunque generalmente incluye las actividades siguientes:

- *Planificación del despliegue* del modelado y/o el conocimiento obtenido.
- *Planificación del control y del mantenimiento*, así, por ejemplo. hay que verificar que el modelo está cumpliendo con las expectativas para las que se ha desarrollado, chequear si el modelo hay que reentrenarlo o sustituirlo por otro modelo, etc.
- *Creación del informe final*, que se puede usar para comunicar los resultados del proyecto y pasos siguientes.
- *Revisión final del proyecto*, donde se establecen las conclusiones finales y se formaliza las lecciones aprendidas para incorporarlas a futuros proyectos de ciencia de datos.

Figura 3.1: Etapas de la metodología CRISP-DM.

Para concluir, subrayar que aunque son varias las metodologías propuestas, existe cierto consenso en que CRISP-DM es la metodología más completa y la más desarrollada y, además, puede ser implementada (como todas las propuestas en la literatura) usando el lenguaje **R** como principal herramienta para la ciencia de Datos.

RESUMEN
El método científico un elemento clave en la definición de ciencia. Bunge (2018) establece una serie de rellas y características para la correcta aplicacion de la metodología. Así como la ciencia se nutre del método científico la ciencia de datos se alimenta de su propipa metodología. Son varías las metodologías propuestas, destacando el CRISP-DM como la más aceptada y utilizada por las emprsas y científicos. CRISP-DP se basa en la organización flexible de seis pilares: entendimiento del negocio, compresión de los datos, preparación de los datos, modelado, evaluación e implementación.

Capítulo 4

Ciencia de datos con R

Emilio L. Cano

4.1. Introducción

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

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

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

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

Uno de los aspectos más espectaculares de **R** es la cantidad de **paquetes** disponibles. Un paquete (*package*) de R es un componente con funcionalidad adicional que se puede instalar en el sistema para ser utilizado por **R**. En el momento de compilar este libro el número de paquetes disponibles en el repositorio oficial es de 18974.

Una vez conocido el mundo de R se plantea la siguiente pregunta, ¿y por qué utilizar **R**? Es imposible dar un único motivo, a continuación se enumeran algunas de las ventajas que tiene utilizar **R**:

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

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

4.2. La sesión de R

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

El interfaz de usuario de R (R GUI, *Graphical User Interface*) cumple las funciones básicas para interactuar con **R**, pero es muy pobre a la hora de trabajar con él. En su lugar, es

[1] https://posit.co, antes RStudio PBC

más conveniento utilizar el entorno de desarrollo RStudio Desktop, (o su versión en la nube https://posit.cloud/) que es como un *envoltorio* del sistema R con más funcionalidades y ayudas, pero manteniendo el mismo nivel de interacción: consola y scripts (archivos de código)[2],[3]. Al igual que **R**, RStudio es una aplicación de software libre, pero en este caso desarrollada y mantenida por una compañía privada, Posit PBC.

4.3. Instalación de R

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

1. Instalación de **R**: http://www.r-project.org
2. Instalación de RStudio: http://rstudio.com

Para completar la instalación de **R**, se muestra cómo instalar[4] algunos paquetes de ejemplo mediante expresiones en la consola o script con la función `install.packages()`:

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

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

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

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

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

4.4. Trabajar con proyectos de RStudio

La manera más eficiente de trabajar con **R**, es mediante proyectos de RStudio. Esto permite abstraerse de los detalles de la sesión de R (espacio de trabajo, directorio de trabajo, *Environment*) ya que al abrir un proyecto, estará todo preparado para seguir el trabajo donde se dejó,

[2] Lo importante es seguir un estilo consistente en cuanto a nombres de objetos, espacios en blanco y distribución del código en el script, véase por ejemplo la guía de estilo de Hadley Wickam (Wickham, 2015).

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

[4] Una vez instalado un paquete no hay que volver a instalarlo.

o empezar de cero si se acaba de crear. Para crear un proyecto de RStudio, desplegamos el menú de proyectos a la derecha en la barra de herramientas y seleccionamos "New Project..." También podemos hacerlo en el menú "File/New Project...".

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

- `datos`: en esta carpeta tendremos los archivos de datos, tanto aquellos orígenes de datos que queramos importar, como los que podamos guardar desde un script.
- `R`: para los scripts. Es posible que solamente tengamos un script en nuestro proyecto, pero si tuviéramos más los podemos meter en esta carpeta.
- `informes`: aquí podemos guardar los archivos Quarto o R Markdown que usemos para generar informes o presentaciones.
- `img`: si en nuestro proyecto vamos a tener imágenes de cualquier tipo, es una buena idea tenerlas en una carpeta aparte.
- `test`: si queremos separar los scripts en los que hacemos pruebas pero no queremos mezclarlos con los "buenos" en la carpeta `scripts`.
- `aux`, `tmp`, `util`, `notas`, doc, ...: este tipo de carpetas vienen bien cuando hay información que está relacionada o es útil para un proyecto, pero no son del proyecto de análisis de datos en sí. Por ejemplo, unas especificaciones de un producto o servicio, un artículo científico, fotografías de una fábrica, comunicaciones con clientes, etc.
- `ejercicios`, `practicas`, ...: si nuestro proyecto forma parte de una asignatura, curso, o similar.

Un aspecto importante cuando se trabaja en proyectos colaborativos es el control de versiones. Este tema se aborda en el Capítulo 46.

4.5. Tratamiento de datos con R

En este apartado se van a empezar a utilizar expresiones de R. Muchas de las expresiones que usamos son llamadas a funciones. Por motivos de espacio no se incluyen mayores explicaciones de las mismas, pero se anima al lector a explorar la ayuda de cada una de ellas para comprender mejor su funcionamiento. La ayuda de cualquier función se puede obtener en la consola usando la expresión `?funcion`, donde `funcion` es el nombre de la función u objeto del que se quiere obtener ayuda.

4.5.1. Estructuras y tipos de datos

Las estructuras y tipos de datos más frecuentes con las que se trabaja en R son:

- **Tablas de datos**: es una colección de variables numéricas y/o atributos organizadas en columnas, en la que cada fila se corresponde con algún elemento en el que se han observado las características que representan las variables. La forma más común es el `data.frame`. Cada columna del `data.frame` es en realidad otra estructura de datos, en concreto, un `vector`. Un ejemplo de data.frame es el conjunto de datos `tempmin_data` del paquete `CDR` que se analiza en el Capítulo 40.

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

Un `data.frame` es un objeto de datos en dos dimensiones, en el que las filas son la dimensión 1, y las columnas la dimensión 2. Podemos "extraer" los datos de un `data.frame` por filas, por columnas, o por celdas. Para extraer una de las variables del `data.frame` se utiliza utilizar el operador `$` después del nombre del data.frame, y a continuación el nombre de la variable.

El operador `<-` asigna al "símbolo" que hay a su izquierda, el resultado de la expresión que hay a su derecha, y lo guarda con ese nombre en el espacio de trabajo. Por ejemplo, la siguiente expresión extrae todas las filas de la columna `tmin` o, dicho de otra forma, el vector con todas las temperaturas mínimas registradas y lo guarda en el objeto `temp_min`.

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

- **Vectores y matrices**: Ya se ha visto que una columna de una tabla de datos es un vector. También se pueden crear vectores con la función `c` y los elementos del vector separados por comas. A modo de ejemplo, la primera de las siguientes expresiones crea un vector llamado `dimensiones` con dos cadenas de texto, y la segunda crea una matriz numérica llamada `coordenadas` a partir de las columnas del data.frame `tempmin_data`.

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

Nótese que la extracción de valores de un data.frame o de una matriz se puede realizar también por sus índices de filas y columnas. En este caso se extraen todas las filas de las columnas 4 y 5.

- **Factor:** es un tipo especial de vector para representar variables de tipo atributo o categóricas. En general, la característica podrá tomar un reducido número de valores diferentes, identificados con etiquetas (*labels*) y que llamaremos **niveles** del factor (*levels*).

Un ejemplo es el dataset `dp_entr` del paquete `CDR` que se analiza en el Capítulo @ref(cap_arboles). La columna `ind_pro11` es un indicador que toma los valores `S` y `N`, mientras que `des_nivel_edu` toma tres posibles valores.

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

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

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

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

- **Fechas**: son un tipo de datos especial que algunas veces provoca problemas al compartir datos entre programas. El `data.frame tempmin_data`tiene la columna `fecha` que puede convertirse de manera inmediata a tipo fecha (`Date`) porque viene en un formato estándar. Véase la ayuda de `strptime` para especificar otros formatos.

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

- **Cadenas de texto**: son estructuras de datos que aparecen como vector de caracteres. La columna `indicativo` del conjunto de datos `tempmin_data` es un ejemplo de este tipo de datos.

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

4.5.2. Importación de datos

En el apartado anterior se han utilizado tablas de datos que están incluidas en el un paquete de R. Pero lo habitual es que se tenga que importar datos para su análisis de fuentes externas, como ficheros. A continuación se describen las formas más habituales[5].

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

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

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

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

- **Texto**: Los **archivos de texto** son el formato más universal para compartir datos. Es también muy común que el equipamiento o el software genere datos en formato de texto. Estos archivos suelen tener extensión `.csv` (*comma separated values*) o `.txt`, aunque puede tener cualquier otro o incluso no tener extensión. Con la siguiente expresión se puede descargar un archivo csv de ejemplo.

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

Si el archivo tiene extensión `.csv` como este vendrá ya con una especificación muy concreta y se pueden usar directamente las funciones `read.csv` o `read.csv2` para tener la tabla de datos en el espacio de trabajo.

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

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

[5] Para poder reproducir los ejemplos, se debe tener una carpeta `data` en el directorio de trabajo.

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

Para saber cómo importar datos desde sistemas gestores de bases de datos ver el Capítulo 6.

4.5.3. Exportación y archivos de datos de R

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

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

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

Tambien se pueden guardar los datos en formato "nativo" de R:

1. Archivos `.RData`: Almacenan un espacio de trabajo entero, y por tanto se pueden guardar varios objetos en el mismo archivo. Al importarlo, los objetos estarán en el espacio de trabajo con su nombre original. Se guardan con la función `save` y se restauran con la función `load`, como en el siguiente ejemplo.

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

2. Archivos `.rds`: Almacenan un único objeto en un archivo. Para importarlos, hay que asignar el resultado al nombre que queramos. Se guardan con la función `writeRDS` y se restauran con la función `readRDS`, como en el siguiente ejemplo.

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

El paquete `haven` de R base y otros paquetes especializados pueden exportar datos a otros formatos de archivo, que no da tiempo a tratar en detalle en este capítulo. Por otra parte, el paquete `rvest`, que forma parte del tidyverse, se puede utilizar para obtener datos de páginas web y otras fuentes de Internet, lo que se suele llamar *web scraping*. Por ejemplo, supongamos que queremos importar la tabla con los datos de comunidades y ciudades autónomas del enlace https://es.wikipedia.org/wiki/Anexo:Comunidades_y_ciudades_autÃşnomas_de_EspaÃśa. Las siguientes expresiones importan esta tabla al data.frame `ccaa_wiki`.

GEMA DICE: Emilio, hay que poner un enlace alternativo o partirlo para que no se salga

```
library(rvest)
url <-
↪  "https://es.wikipedia.org/wiki/Anexo:Comunidades_y_ciudades_autónomas_de_España"
ccaa_wiki <- url |>
  read_html() |>
  html_node(xpath = "/html/body/div[3]/div[3]/div[5]/div[1]/table") |>
  html_table(fill = TRUE)
```

Alternativas, por ejemplo:

```
url <- "https://es.wikipedia.org/wiki/Anexo:Tabla_peri%C3%B3dica"
url <- "https://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses"
```

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

4.6. Organización de datos con el *tidyverse*

4.6.1. El *tidyverse* y su flujo de trabajo

El *tidyverse* es, según se define en su propia página web, "*an opinionated collection of R packages designed for data science*". Es decir, un conjunto de paquetes de R "opinables" diseñados para ciencia de datos. Las principales ventajas (opinables) de utilizar el *tidyverse* son tres:

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

Uno de los paquetes más populares del `tidyverse` es `ggplot2`, que proporciona una "gramática de gráficos" (Wickham, 2016), y es pieza clave en el `tidyverse` actual, junto con los paquetes `dplyr` y `tidyr`. El flujo de trabajo propuesto por el *tidyverse* se describe en el libro "R for Data Science" (Wickham and Grolemund, 2016), y se sintetiza en la figura 4.1.

Figura 4.1: Flujo de trabajo en Data Science (fuente: R for Data Science, ver bibliografía)

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

Dentro del flujo de trabajo de la figura 4.1, ya se ha tratado la primera etapa (*Import*) en el apartado 4.5.2. Es importante señalar que, al utilizar las funciones del *tidyverse*, los datos se organizan en objetos de clase `tibble`, que es una extensión del `data.frame` de R base. Las principales diferencias son:

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

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

Siguiendo con el esquema de la figura 4.1, en este apartado se verán las etapas *Tidy* (organizar) y *Transform* (transformar), mientras que la visualización (*Visualise*), modelización (*Model*) y comunicación (*Communicate*) se verán en otros capítulos del libro. Una de las características de la forma en que están programados los paquetes del *tidyverse* es que se puede trabajar[6] con *pipes*.

[6]Existe una guía de estilo del *tidyverse*, que se puede consultar en https://style.tidyverse.org. Hay incluso una serie de Addins en RStudio para comprobar y aplicar esta guía de estilo a través del paquete `styler`.

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

```
lado_izquierdo |> lado_derecho
```

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

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

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

No obstante la ventaja de usar los pipes es que se pueden encadenar, de forma que el resultado de cada operación pasa a la siguiente expresión del *pipeline*, como en el siguiente ejemplo:

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

4.6.2. Transformación de datos con `dplyr`

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

4.6.2.1. Operaciones con filas

Los verbos definidos para estas operaciones son:

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

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

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

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

- **`slice()`**: selecciona por el índice de la fila. También hay una serie de funciones "asistentes" (*helpers*) para obtener los índices que se utilizan con frecuencia. Por ejemplo:
  - `slice_head()` y `slice_tail()` obtienen las primeras y últimas filas respectivamente (por defecto, una). Se puede especificar `n` o `prop` (proporción de filas)
  - `slice_sample()` obtiene una muestra aleatoria de `n` filas (o proporción `p`)
  - `slice_min()`, `slice_max()` obtienen las filas que contienen los menores o mayores valores respectivamente de la variable indicada en el argumento `order_by`. Si no se especifica `n` o `prop`, solo los que coinciden con el mínimo o máximo. Nótese que puede haber más de una fila que cumpla la condición.

Véase el resultado de los siguientes ejemplos:

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

4.6.2.2. Operaciones con columnas

Los verbos definidos para estas operaciones son:

- **`select()`**: indica cuando una columna es incluida o no.

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

En cuanto a la **modificación** de datos tenemos múltiples posibilidades.

- **`rename()`**: cambia el nombre de la columna.
- **`mutate()`**: cambia los valores de las columnas y crea nuevas columnas. `transmute` funciona igual que `mutate`, pero la tabla de datos resultante solo contiene las nuevas columnas creadas.
- **`relocate()`**: cambia el orden de las columnas.
- **`summarise()`**: Resume todos los datos en una sola fila usando alguna función de resumen, por ejemplo la media, desviación típica, etc.

```
pm10 |>
  rename(zona_calidad_aire = zona)
pm10 |>
  relocate(fecha, .before = estaciones)
pm10_na <- pm10 |>
  mutate(isna = is.na(daily_mean))
pm10_media <- pm10 |>
  summarise(Media = mean(daily_mean, na.rm = TRUE))
```

En este punto es importante señalar que dentro de la función `mutate` se puede usar cualquier función vectorizada para transformar las variables. Por ejemplo, se podría transformar una columna con las funciones `as.xxx` que se vieron en el apartado 4.5.1, aplicar formatos a fechas o usar funciones del paquete `lubridate` para trabajar con este tipo de datos. A medida que se avance en el libro irán apareciendo aplicaciones que ahora quizás no sean tan evidentes.

4.6.2.3. Operaciones de resumen y agrupación

La primera operación de resumen que puede surgir es "contar" filas. La función `tally()` devuelve el número de filas totales de un `data.frame`. La función `count()` nos puede dar también este número, pero si le pasamos como argumento además alguna variable, nos da el número de filas para cada valor diferente de dicha/s variables. Estos recuentos se pueden añadir a la tabla de datos con las funciones `add_count()` y `add_tally()`, lo que permite calcular frecuencias absolutas y relativas fácilmente.

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

La función `summarise()` (o , equivalentemente, `summarize()`) aplica alguna función de resumen a la/s variable/s que se especifiquen. El paquete `dplyr` tiene algunas funciones de resumen adicionales, como `n()` (número de filas), `n_distinct()` (número de filas con valores distintos), `first()`, `last()`, `nth()` (primero, último, y enésimo, respectivamente).

En muchas ocasiones las operaciones de análisis se realizan en grupos definidos por alguna variable de agrupación. La función `group_by()` "prepara" la tabla de datos para realizar operaciones de este tipo. Una vez agrupados los datos, se pueden añadir operaciones de resumen como las vistas anteriormente. A veces hay que "desagrupar" los datos, para lo que utilizaremos la función `ungroup()`.

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

```
contam_mad |>
  filter(nom_abv == "NOx") |> # filtramos por PM10
  group_by(zona) |>
  summarize(
    min = min(daily_mean, na.rm = TRUE),
    q1 = quantile(daily_mean, 0.25, na.rm = TRUE),
    median = median(daily_mean, na.rm = TRUE),
    mean = mean(daily_mean, na.rm = TRUE),
    q3 = quantile(daily_mean, 0.75, na.rm = TRUE),
    max = max(daily_mean, na.rm = TRUE)
  )
#> A tibble: 5 × 7
    zona            min    q1 median  mean    q3   max
    <chr>         <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
#> 1 Interior M30 0.0833  32.4   54.1  72.9  90.0  759.
#> 2 Noreste      1       23.8   39.6  56.2  68.9  516.
#> 3 Noroeste     0       12.0   20.3  29.7  34.5  352.
#> 4 Sureste      0       29.1   45.4  64.6  77.2  453
#> 5 Suroeste     0.667   33.5   59.6  90.5 114.   666.
```

4.6.3. Combinación de datos

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

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

Las funciones requieren como argumentos dos tablas de datos y la especificación de las columnas coincidentes. Si no se especifica, hace las uniones por todas las columnas coincidentes en ambas tablas. Para las filas que solo están en una de las tablas, se añaden valores `NA` donde no haya coincidencias.

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

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

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

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

4.6.4. Reorganización de datos

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

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

- `pivot_longer()`: el argumento `names_to` asigna el nombre de la nueva variable que va a indicar de qué columna vienen los datos; y el argumento `values_to` asigna el nombre de la nueva variable que va a contener el valor de la tabla original.

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

Las funciones `pivot_longer()` y `pivot_wider()` admiten otros argumentos `names_xx` y `values_xx` para personalizar la forma de reestructurar los datos. En la mayoría de las ocasiones será suficiente con las explicadas (`xx_from` y `xx_to`), consulta la ayuda de las funciones en caso necesario, o el [artículo sobre *pivoting*](https://tidyr.tidyverse.org/articles/pivot.html.

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

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

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

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

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

- `drop_na()` del paquete `tidyr` permite eliminar las filas que tienen valores perdidos en ciertas variables (o en cualquiera si no se especifica ninguna).

- `replace_na()` susituye los valores perdidos en cada variable por el valor especificado.

- `fill()` permite "rellenar" valores perdidos con los últimos encontrados.

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

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

RESUMEN

- **R** es software libre y gratuito, mantenido por una enorme comunidad.
- La forma de interactuar con **R** es mediante expresiones, que se escriben en *scripts*, y al ejecutarlas se obtienen los resultados.
- La sesión de **R** se inicia en un directorio de trabajo.
- Los objetos de datos que se vayan a usar deben estar en el espacio de trabajo.
- RStudio es un "envoltorio" de **R**, y por tanto **R** tiene que estar instalado en el sistema para poder usar RStudio.
- Los paquetes se instalan una sola vez, y lo podemos hacer desde el panel *Packages* o con la función `install.packages` en la consola.
- Para acceder a las funciones de los paquetes, deben cargarse antes con la función `library` o usar la sintaxis explícita `paquete::funcion()`
- La tabla de datos o `data.frame` es la estructura más adecuada para análisis de datos.
- Cada característica estudiada es un `vector`, que normalmente se almacena como una columna de la tabla de datos.
- Los elementos de un vector son todos del mismo tipo.
- Las variables categóricas se representan mediante objetos de tipo `factor`, con niveles y etiquetas.
- Las listas contienen colecciones de objetos que pueden ser de distintos tipos.
- El paquete `readxl` del `tidyverse` importa archivos de Excel.
- Los ficheros de texto más populares son los `.csv`.
- La exportación a ficheros de texto se realiza con las funciones `write.*` de forma similar a como se importan. Para guardar archivos de excel usamos el paquete `openxlsx`.
- Los datos del espacio de trabajo se pueden guardar en archivos `.RData` o `.rds`.
- El *tidyverse* es un conjunto de paquetes que facilita las tareas de análisis de datos.
- El operador *pipe*, `|>` permite "pasar" valores a funciones de forma encadenada.
- Las operaciones básicas con una tabla son filtrado, selección y resumen.
- Para crear nuevas columnas en las tablas de datos usamos la función `mutate`.
- Para combinar tablas con columnas comunes se usan las funciones `xx_join`.
- Los datos de una tabla se pueden apilar y extender para tener datos "anchos" o "largos".
- Cuando tenemos datos perdidos, podemos optar por eliminar las filas o imputar los valores.

Capítulo 5

Ética en la ciencia de datos

Mónica Villas

5.1. ¿Por que la ética en la ciencia de datos?

Los primeros documentos de ética se remontan a los tiempos de Platón, cuando se comenzó a pensar más allá del día a día y a establecer una relación causa-efecto. Este manual se va a centrar en la ética aplicada, que es la utilización de la ética en la práctica. Algunos ejemplos de ética aplicada pueden ser la ética profesional, la bio-ética, la ética del medio ambiente y así un sinfín de ellas más. Si se refiere al trabajo concreto de un científico de datos, se debe tener en cuenta los pasos concretos desde que se enuncia el problema a resolver hasta que se diseña un algoritmo para resolverlo. El científico de datos tiene que tomar decisiones en cada uno de los pasos del proceso: la recopilación de datos, la transformación, la definición de objetivos a medir, el uso de algoritmos y finalmente la explicación de los resultados. En todos estos pasos, tiene que usar su pensamiento crítico y tomar decisiones que pueden ser correctas o no, en lo que a la ética se refiere. En la parte de automatizar un proceso concreto, y encontrar el algoritmo que ayude a automatizarlo, es otro punto clave en la toma de decisiones. Si este paso no se hace de manera adecuada, la toma de decisiones automáticas puede "perpetuar" algunos de los problemas éticos, como por ejemplo los sesgos, o se pueden tener decisiones tomadas por los algoritmos que el propio científico de datos no puede explicar, cómo se verá más adelante. Para ver esto con algo más de detalle, siempre es mejor usar un ejemplo. Si se toma un ejemplo de automatización de contrataciones, en el que un científico de datos necesita desarrollar un algoritmo para que contrate a los mejores profesionales para su compañía, estos serían los pasos a seguir. Primero tiene que entender que significa los mejores profesionales y definir los atributos que los representan. Después, tiene que buscar datos históricos de la compañía, recopilar éstos y estar seguros qué esos datos cumplen con la normativa de privacidad establecida, especialmente si la compañía reside en Europa. Aquí, el científico de datos, debe pensar en temas como la procedencia de los datos, ¿Cuál es la fuente?, ¿a quién pertenecen los datos?, ¿están los datos anonimizados para que no podamos identificar a una persona de manera unívoca? y algunas

preguntas similares referidas a la privacidad. Seguidamente, se tendrá que revisar la muestra de datos y asegurar que se tiene una muestra de datos cuyos atributos (edad, profesión, experiencia, raza, género, procedencia geográfica, ...) no incluyen sesgos, es decir, no se tiene por ejemplo muchos más casos de personas de color que de raza caucásica , o más personas mayores que menores, o más mujeres que hombres. En definitiva, que la muestra que tiene es suficientemente representativa de la población con la que va a trabajar. Sino, un científico de datos sabe cómo utilizar técnicas para resolver este tipo de problemas que le van a permitir tener una muestra de datos balanceada. Además, se ha de tener cuidado con los datos personales, referentes a las personas como género, edad, raza , ...etc dado que la toma de decisiones no puede tener en cuenta estos atributos porque se estaría discriminando y por tanto no actuando de manera ética. Como se puede ver en este sencillo ejemplo, en su día a día el científico de datos tiene que tomar decisiones, no sólo técnicas, que influyen en el resultado de su trabajo y que pueden afectar a otras personas. Generalmente los científicos de datos suelen ser profesionales que provienen del mundo técnico, de carreras como ingeniería o matemáticas y que en general sus conocimientos de ética son más escasos. De ahí que sea clave su conocimiento de los fundamentos básicos para incorporarlos en su día a día.

Al igual que para los médicos existe un juramento hipocrático no existe una guía común para el científico de datos, donde se describa como debe comportarse. Uno de los documentos que podría servir de base se refiere a las acciones de los profesionales de informática que publica la asociación ACM.

5.2. Los principios éticos

Desde inicios del 2010, el boom de la IA (Inteligencia Artificial) ha sido exponencial, debido a la gran cantidad de datos así como la mejora en la computación y la disponibilidad de un gran número de algoritmos. Por ello la demanda de científicos de datos que sean capaces de transformar estos datos en información clave para las empresas, ha crecido mucho en los últimos años. Asimismo, desde 2016, distintos organismos, asociaciones, empresas, gobiernos han publicado numerosos documentos, donde se resalta que es clave la necesidad de principios éticos de la IA. Google, IBM, Amazon publicaron sus principios éticos y otro ejemplo conocido es la publicación del tratado de Asimolar basado en los principios de Asimov. La mayoría de estos documentos están desarrollados por perfiles multidisciplinares: científicos de datos, abogados o expertos en ética, que resaltan la importancia de tener en cuenta los principios éticos en la toma de decisiones automáticas cuando se utiliza la ciencia de datos. Un principio no es ni más ni menos que "aquello que permite preservar los derechos y libertades de las personas, sin frenar la innovación tecnológica" Olmeda and Ibánez (2022). La mayoría de los principios se pueden agrupar en cuatro grandes categorías: **autonomía, justicia, evitar daños y generar beneficios.** Algunos ejemplos deprincipios que se pueden agrupar en alguna de estas categorías son: **transparencia, explicabilidad, privacidad, accesibilidad o equidad.** No hay un acuerdo a nivel mundial, sobre cuáles deberían ser los principios claves de la IA, pero sí que se están desarrollando proyectos supranacionales como el de la UNESCO donde recientemente se ha firmado un acuerdo sobre la ética de la IA, con todos los países miembros.

Uno de estos ejemplos es la regulación europea publicada en Abril de 2021 y que ha sido fruto del trabajo de tres años. Liderado por ungrupo de expertos HLEG AI (High Level Expert Group)

y con un enfoque consultivo, con el que se han realizado distintas publicaciones desde 2018, mientras que se recogían los comentarios de la sociedad civil, instituciones públicas, empresas e instituciones académicas.

La propuesta de regulación, publicada en abril de 2021, ha elegido un enfoque de riesgos, que se centra en clasificar los riesgos en cuatro tipologías:

- **Riesgos inaceptables**, como el uso de aplicaciones de *social scoring* o para temas judiciales.
- **Riesgo alto**, como el uso de aplicaciones de contratación o médicas que deberán ser supervisadas por organismos designados antes de su publicación.
- **Riesgo medio** en las que será necesaria incluir la explicabilidad necesaria para entender la toma de decisiones del algoritmo
- **Riesgo bajo** para cualquier otro tipo de aplicación

En el resto del mundo, aún el progreso en este tipo de regulación está siendo algo más lenta, aunque países como Estados Unidos, que hasta ahora no había puesto el foco en este tipo de regulaciones está empezando a trabajar en ello desde finales del año pasado. Por otro lado, China conocida mundialmente por su falta de respeto a la privacidad, está empezando a dar algún paso en esta área. Según un artículo de Forrester, China está empezando a cambiar su política en este sentido, y como ejemplo en Marzo de 2022 ha lanzado una regulación donde las empresas tienen que informar mejor a los usuarios sobre sus algoritmos de recomendación. En definitiva, parece que la necesidad de la ética para proyectos de ciencia de datos está avanzando poco a poco en todas las geografías, y Europa es por el momento un ejemplo a seguir.

5.3. La importancia de los sesgos

En primer lugar, se debería establecer la definición de sesgo, el sesgo es dar un peso desproporcionado a favor o en contra de algo. Si revisamos la bibliografía al respecto se pueden encontrar multitud de tipos de sesgos pero en lo que respecta a la ciencia de datos, los sesgos de los que vamos a hablar son los tipos de sesgos algorítmicos, y algunos de los ejemplos más conocidos. Este sesgo algorítmico sucede, cuando en el proceso del desarrollo de un sistema de inteligencia artificial, se incluyen en alguna parte del proceso. Se puede observar el sesgo en la adquisición de los datos, partiendo de muestras que ya tienen un sesgo o que no están balanceadas, también puede ocurrir este sesgo en base a la selección de los atributos que van a formar parte del modelo, así como en el despliegue del modelo si el contexto en el que se despliega el algoritmo es diferente de donde se despliega como refleja la figura 1 .Los estudios de estos sesgos algorítmicos están enfocados a evitar que se aumenten o perpetúen sesgos de cualquier tipo, teniendo en cuenta que los algoritmos tienen como objetivo generalizar. Como se veía en la sección anterior, mucha de la regulación que se está desarrollando en Europa va enfocado a **mantener el principio de equidad**, es decir, a no tener decisiones sesgadas.

Figura 5.1: Sesgos en el proceso de machine learning - fuente(IBM)

A continuación, se describe algún ejemplo en más detalle de cómo los algoritmos están tomando decisiones e incluyendo sesgos.

El ejemplo de COMPAS (*Correctional Offender Management Profiling for Alternative Sanction*) es una aplicación que da soporte al sistema de justicia americana y que decide si la persona que va a ser juzgada tiene la probabilidad de ser reincidente o no. Cada acusado tenía asignado tres riesgos: Riesgo de reincidir, Riesgo de violencia y Riesgo de aparecer en el juicio. El índice de riesgo se establecía de 1 a 4 como bajo, de 5 a 7 como medio y de 8 a 10 como alto. Si la persona puede ser reincidente espera a que ocurra el juicio en la cárcel, y sino no tiene que ir a la cárcel hasta que secelebre el juicio. Diversos estudios y organizaciones analizaron los datos y no parecía que hubiera un problema de sesgo inicialmente. Sin embargo, la organización PROPUBLICA, recogió datos de unas 12.000 personas, durante 2013 y 2014 y demostró que la aplicación estaba sesgada. El proceso que se siguió fue el siguiente:

- Partiendo del proceso de asignación de un riesgo se construyó el historial delictivo del acusado
- Para determinar la raza, se usó la clasificación establecida, de negros, blancos, hispanos y asiáticos
- Se revisó la definición de reincidencia, y como se establecían los riesgos en la aplicación de COMPAS
- Solamente analizaron los riesgos para "reincidencia" y "reincidencia violenta"
- Se analizó el índice de reincidencia y de violencia en dos años y su distribución respecto a la raza
- Para comprobar la disparidad entre la raza y el índice de riesgo, se utilizó una regresión logística que consideraba la raza, la edad, la historia criminal, la reincidencia futura, el grado de los cargos y el género.
- Par ver la exactitud del algoritmo se usó un modelo de Cox
- Se utilizó una muestra de unos 7300 acusados (para los que se tenía datos de 2 años) para analizar la tasa de falsos positivos y falsos negativo

Las personas de color tenían un índice de riesgo de reincidencia mucho más alto que las personas de raza caucásica. La herramienta predecía bien en el 60 % de los casos estudiados el riesgo de reincidencia, pero sólo en el 20 % de los casos lo hacía de manera correcta en el riesgo de reincidir de manera violenta. Se incluye un resumen de las conclusiones en la siguiente tabla:

Casuística en el estudio con datos de 2 años	Resultados en porcentaje
Los acusados de raza negra se les asignaba un riesgo más alto de reincidencia que los de raza negra	45 % a los de raza negra 23 % a los de raza caucásica
Los acusados de raza blanca se les asignaba un riesgo más bajo de reincidencia que los de raza negra	48 % a los de raza blanca 28 % a los de raza negra
Mayor asignación de riesgo de reincidencia a las personas de color	77 % más de riesgo de reincidir a las personas de raza negra que a los de raza blanca
Se determinó que las variables que tenían mayor importancia para la asignación de riesgo de reincidencia era la edad, la raza y el género	< 25 años tenía 2.5 veces más de probabilidad de ser asignado un riesgo alto 45 % si además eran de raza negra Casi un 20 % si la persona era mujer

Los detalles se pueden encontrar en este repositorio de github que solamente incluye el código en Python.

Hay multitud de ejemplos publicados, respecto al tema de sesgos, una de las mejores referencias sobre sesgos es el libro O'neil (2016)

5.4. ¿Es necesaria la explicabilidad?

La explicabilidad es otro de los principios clave de la IA ética, y además ha sido seleccionado por Europa como uno de los ejes de su propuesta de regulación de IA ética.

XAI (*Explainable AI*) es un término que acuño DARPA en el año 2017 y que lo que hace es agrupar no sólo el concepto de interpretabilidad para los algoritmos de machine learning sino la parte de la psicología que tiene que ver con las explicaciones como se puede ver en la figura 2. No sé trata solamente de entender la toma de decisión del algoritmo sino de explicarlo de manera adecuada dependiendo del tipo de usuarios y estar seguros que la persona que recibe las explicaciones lo ha entendido. Si se toma el ejemplo de un algoritmo que selecciona imágenes cuando contienen un posible tumor, no serán las mismas explicaciones las que necesitará un científico de datos o un médico. Para el científico de datos será mucho más útil revisar los métricas propias del algoritmo (exactitud, precisión, sensibilidad, ..etc) y además saber cuáles de los atributos de entrada del algoritmo han tenido más peso en la decisión. En cambio, al médico lo que le interesará será una explicación menos técnica, más cualitativa, en la que se le explique con detalle por ejemplo porque se seleccionó esa muestra frente a otras, mencionando el tamaño, la forma u características que sean mejor entendidas por el profesional médico.

Figura 5.2: Explicabilidad según Darpa

Los algoritmos pueden clasificarse en algoritmos de "**caja blanca" o transparente**, aquellos que son fácilmente interpretables, y algoritmos de **opacos o de "caja-negra"** aquellos que no son interpretables y que requieren de herramientas adicionales para ello. Normalmente se tiene que establecer un balance entre la interpretabilidad y la exactitud dado que son métricas con una relación inversamente proporcional. A mayor exactitud menor interpretabilidad y viceversa. Los algoritmos que son más interpretables son aquellos más sencillos como los algoritmos de clasificación, regresión lineal o los árboles de decisiones. Otros como los modelos de *Random forest*, XGboost o algoritmos de *Deep learning* son mucho más exactos pero no son tan interpretables y esto puede presentar ciertos problemas a la hora de usarlos en la toma de decisiones en las compañías, dado que debo ser capaz de explicar la decisión del algoritmo.

Figura 5.3: Explicabilidad vs Exactitud

5.5. Sesgos y explicabilidad en R

Para terminar, es muy relevante conocer las herramientas opensource o comerciales disponibles para que se puedan usar en análisis de sesgos o explicabilidad. Las herramientas son relativamente recientes, han ido surgiendo desde el 2018 y van evolucionando rápidamente.

En el caso de las herramientas para detectar el sesgo los proveedores son Microsoft, IBM, Google, Aequitas, Pymetric, Linkedin y el resto Opensource. La mayoría de ellas están abiertas

a contribuciones externas y todas ellas utilizan mecanismos para la detección de sesgos, aunque solamente la de Microsft e IBM incluyen algoritmos para la mitigación de estos sesgos.

Referente a las herramientas sobre explicabilidad los proveedores más relevantes son Google, IBM, Oracle, H20 , el resto son opensource. La mayoría de ellas se pueden usar con algoritmos de caja blanca o caja negra. Respecto a los tipos de explicaciones para distintos usuarios sólo una de ellas por el momento, la herramienta de IBM incluye esta funcionalidad. Se puede resaltar también, la facilidad con la que H20 permite elegir el nivel de exactitud y explicabilidad en el momento del diseño del algoritmo. Para un mayor detalle se puede consultar dos tablas comparativas que se incluyen en el Olmeda and Ibánez (2022)

Recursos en R para equidad:

- Tutorial de fairness (2021): Este tutorial explica las distintas métricas usadas para medir la equidad (paridad demográfica, paridad proporcional, paridad predictiva. . . etc) y permite crear la distintas métricas y visualizarlas. El tutorial usa losdatos de COMPAS.
- Librerias de R incluidas en IBM fairness360 (2020): Este paquete incluye algoritmos para detectar el sesgo, pero también para mitigarlo.
- Libreria EDFFair (2022) tiene una aproximación distinta, dado que permite al usuario ajustar el nivel de equidad frente a la exactitud, y poder mantener el equilibrio requerido. Esto se explica en detalle en Matloff and Zhang (2022)

Recursos en R para explicabilidad:

- Algunas de las herramientas más conocidas en explicabilidad, que merecen mención aparte son SHAP y LIME, disponibles en R y en Python y usadas en muchos paquetes comerciales. **SHAP (2018)** es uno de los métodos más usados para la explicabilidad. Utiliza los valores de Shapley para poder explicar cualquier tipo de modelo, los detalles se pueden encontrar en Aas et al. (2021) donde también se proporciona el código. **LIME (2017)** es otro de los métodos usados para explicabilidad, trata de ajustar un modelo local alrededor de un punto concreto y lo que hace es estudiar las perturbaciones alrededor de este modelo. Se puede encontrar la explicación detallada en **?** donde también se proporciona el código

- El artículo Matloff and Zhang (2022) hace una recopilación de 27 librerías de R, incluyendo LIME y SHAP y el código en github para usar cada una de ellas.

- Dalex(2022) es un paquete disponible en R de reciente creación para ayudar a crear explicaciones partiendo de un modelo, y también proporciona el código necesario.

  Este área está evolucionando mucho en los últimos años, y están surgiendo multitud de técnicas nuevas alrededor de la explicabilidad, que van a permitir entender mejor el proceso de decisión realizada por algoritmos más complejos.

Parte II

Bienvenido a la jungla de datos

Capítulo 6

Gestión de bases de datos relacionales

Ismael Caballero y Ricardo Pérez del Castillo

6.1. Introducción

El mundo real en el que estamos inmersos es puramente analógico: está lleno de **entidades** que se relacionan entre ellas o consigo mismo a través de unos determinados eventos que representan determinados **hechos**. Tanto **entidades** como **hechos** tienen una colección de características observables (normalmente llamados **atributos**), que pueden ser de interés en el contexto de una determinada aplicación.

Para estas aplicaciones que demandan el uso de datos del mundo real, es preciso realizar **observaciones** de esos **atributos** de las **entidades** y de los *hechos* que son relevantes. Al conjunto de entidades y hechos del mundo real que son relevantes para una aplicación se les conoce como **Universo del Discurso** (Piattini et al., 2006). Es importante capturar la semántica del Universo del Discurso mediante los modelos correspondientes.

Es preciso, por tanto, asimilar que los datos que son de interés para una determinada aplicación deben ser capturados mediante un **proceso de observación y digitalización** de los valores de los atributos relevantes de las entidades y hechos del mundo real. Durante el proceso de observación se pueden producir errores que pueden derivar en problemas relacionados con la calidad de los datos (Price and Shanks, 2004). Por ejemplo, supóngase que las observaciones requieren una determinada frecuencia mínima de observación en relación con la velocidad en la producción de los hechos; si esta frecuencia no es adecuada, entonces, la cantidad de observaciones realizada será insuficiente para modelar el hecho, llevando a un estado inconsistente entre lo sucedido y lo observado.

Una vez capturados estos datos, pueden ser usados en los procesos de negocio o analizados para realizar una tarea o para producir un conocimiento del mundo real que hasta ahora no se tenía

visión o conocimiento del mundo real (siempre teniendo en cuenta que no dejan de ser sino una representación del mismo, y que por tanto hay que tener mucho cuidado al extrapolar los resultados de dichos análisis) (Davenport and Harris, 2017).

6.2. Concepto de Base de datos

Para poder habilitar esos análisis de forma automática mediante las potentes técnicas tratadas en el resto de capítulos de este libro, es necesario poder almacenar los datos en algún lugar como `repositorios` o `bases de datos`, donde se puedan fácilmente añadir, borrar, recuperar o modificar los datos.

De acuerdo con Piattini et al. (2006), una **base de datos (BD)** es una **colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada**. En una base de datos, los valores correspondientes a los atributos que han de ser compartidos por diferentes usuarios y aplicaciones deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos. Este modelo debe captar las interrelaciones y restricciones existentes en las entidades y hechos del mundo real al que representan. Existen diferentes tipos de modelos que permiten estructurar y representar la semántica de los datos.

En el ámbito de los Sistemas de Información, se han desarrollado programas que dan soporte a todo el proceso de creación y explotación de las bases de datos. A estos programas se les conoce como **Sistemas Gestores de Bases de Datos (SGBD)**. Como ejemplos de estos SGBD se pueden citar Microsoft Access, Microsoft SQL Server, Oracle Server, MySQL, MariaDB, Informix, MongoDB... En cualquier caso, como se verá más adelante en este capítulo, los SGBD que han venido siendo más amplicamente utilizados hasta ahora son los conocidos como **relacionales (SGBDR)**, aunque con el auge de Big Data y de Machine Learning, esta tendencia está cambiando y empiezan a desplegarse cada vez más SGBD conocidos como **NoSQL** (véase capítulo 7). Para evitar confusiones, se hace importante diferenciar entre lo que sería la base de datos propiamente dicha (**como una colección de datos almacenada en un fichero de datos**), y el software SGBD específico, ya sea relacional o NoSQL: una misma base de datos, con las correspondientes adaptaciones, puede ser gestionada usando diferentes SGBD. Habitualmente, los tipos de SGBDR más usadas son los que tienen capacidades multiproceso/multiusuario, ya que permiten acceder a datos compartidos mediante el uso de interfaces de datos para ejecutar diferentes tipos de análisis empleando lenguajes de programación más potentes - o versátiles - como R o Python.

6.2.1. Gestión de los datos en una base o repositorio de datos

Las organizaciones usan datos para sus procesos de negocio. En función de esto es posible diferenciar entre diferentes tipos de datos (Mahanti, 2019) como los recogidos en la tabla mostrada en Fig. 6.1:

[**GEMA!!**: como no he conseguido disminuir el tamaño de la letra en la tabla con RMarkDown, he puesto una imagen con la tabla, creo que ahora queda más o menos bien. PEro el problema es que en el caption aparece la palabra "figura" y no "tabla"]

Tipos de Datos	Amplitud de los datos	Volumen	Frecuencia Actualización	Número de Columnas	Ejemplos	Propietarios o Stewards	Sistema Típico de Registro
Metadatos	Organización	Muy baja	Casi estática	Varias	Definiciones de atributos	Organización	Repositorio de Metadatos
Datos de Referencia Core	Organización, o casi toda la organización	Baja	Muy Baja	Pocas	SIC, Países, Códigos de Divisas, Esquemas ISSO	Normalmente externos a la organización	Repositorio de datos de referencia corporativo
Datos Maestros Comunes	Múltiples sistemas abarcando la organización	Medio	Bajo	Muchas	Cliente, Empleado, Producto, Cuentas Bancarias	Principalmente usuarios que lo crean o lo necesitan	Varios escenarios
Datos Transaccionales	Transacciones actuales	Alto	Medio	Muchas	Compras, inversiones, ...	Sistema que generaron las transiciones	Sistema que generaron las transiciones

Figura 6.1: Tipos de datos

Para poder usar estos datos como parte de los procesos de negocio, es necesario desarrollar aplicaciones que realicen los siguientes cuatro tipo de operaciones (normalmente conocidas como `operaciones CRUD`):

1. **Crear datos (*Create*)**, inserta datos en el repositorio de datos.
2. **Leer datos (*Read*)**, recupera datos del repositorio para aprovisionar el proceso de negocio, o bien para realizar alguna operación específica.
3. **Actualizar datos (*Update*)**, modifica el valor de los atributos correspondientes a los hechos o entidades para actualizarlos a nuevas observaciones
4. **Borrar datos (*Delete*)**, elimina en bloque o selectivamente los datos almacenados en el repositorio de los datos.

En cualquier caso, estos procedimientos de inserción, actualización, recuperación, y borrado deben garantizar siempre la seguridad del conjunto de los datos de modo que sólo sean accesibles por aquellos usuarios que estén autorizados a trabajar con ellos teniendo en cuenta el propósito establecido para los datos (Piattini et al., 2006).

La forma de implementar estas operaciones depende fuertemente del formato (modelo lógico) en el que estén almacenados los datos. Aunque existen diferentes modelos (estructurados, semi-estructurado, no estructurados), en este capítulo, el enfoque se hace sobre el modelo relacional desarrollado por Codd (Codd, 1970), ya que es el más ampliamente usado en el ámbito empresarial y es el que implementan los SGBDR. Para poder dar soporte a las operaciones CRUD anteriormente citadas en bases de datos relacionales, se desarrolló un lenguaje llamado `Lenguaje Estructurado de Consulta` (**Structured Query Language**, `SQL`), que se verá en la siguiente sección.

6.3. El Lenguaje Estructurado de Consulta (SQL)

Los principios de SQL están establecidos en el estándar internacional ISO/IEC 9075:1989 como un mecanismo para identificar y regular las expresiones necesarias que permiten manejar bases de datos relacionales. Al ser un estándar, es importante recordar cada fabricantes de SGBDR como Oracle con su Oracle Database Manager Server o con MySQL, Microsoft con su SQL

Server, IBM con su DB2... implementan en sus productos su propia versión del estándar SQL. Y aunque son prácticamente iguales, hay ligeros matices que les permiten diferenciarse de la competencia y que por tanto deben ser conocidos cuando se utilicen los correspondientes productos comerciales. No obstante existen en el mercado algunas soluciones *open source* como MariaDB o PostgreSQL. En este capítulo, todos los ejemplos que se han desarrollado trabajan contra un servidor MySQL 8.

SQL tiene diferentes tipos de sentencias o instrucciones que dan soporte a los diferentes aspectos de las interacciones con la base de datos. No obstante, cualquier manual de SQL permitirá tratar en profundidad todos los elementos sintácticos del lenguaje, pero es importante recordar que los detalles específicos de la sintaxis específica dependerá fuertemente del SGBDR empleado. Aunque al final de un libro se añade una plantilla con un resumen de la sintaxis de SQL, se recomienda, para los ejemplos propuestos en este libro el manual de referencia de SQL de MySQL. En las siguientes secciones se da una visión global de dichos grupos de sentencias

6.3.1. SQL como Lenguaje de Definición de Datos (LDD)

El elemento central del modelo relacional es la **relación**, cuya implementación se realiza a través de una **tabla de datos**. Así, SQL se utilizará para definir todos los elementos necesarios para crear y modificar las tablas de datos. Se tienen tres tipos de instrucciones básicas para gestionar las tablas como estructuras de datos:

- ***Create***, permite crear un componente de la base de datos, tal como una base de datos, una tabla, una vista,... En el siguiente ejemplo, se crea usando instrucciones SQL, primero una base de datos llamada `Biblioteca` y luego se crea una tabla `Autor` con seis atributos (`CodAtutor`, `Nombre`, `Apellido1`, `Apellido2`, `Pseudonimo`, y `Nacionalidad`), donde se podrán almacenar valores correspondientes a dichos atributos, conformando así la base de datos:

```
create database Biblioteca
 create Table Autor (
   CodAutor nvarchar (20) primary key,
   Nombre    nvarchar(40) not null,
   Apellido1 nvarchar(50) not null,
   Apellido2 nvarchar(50),
   Pseudonimo nvarchar(50),
   Nacionalidad nvarchar (50)
 );
```

- ***Alter***, permite modificar la estructura de un componente, añadiendo por ejemplo, atributos a una tabla, restricciones a un atributo, o modificando el tipo de datos de algún atributo existente; también para eliminar un atributo de una tabla existente. Siguiendo el ejemplo anterior, con la siguiente instrucción se añade un nuevo atributo `LocalidadNacimiento` a la tabla `Autor`:

```
alter table Autor add LocalidadNacimiento nvarchar(50) not null
```

- ***Drop***, esta instrucción sirve para eliminar un componente específico, como por ejemplo una tabla, una vista,... Pero no sirve para eliminar los valores almacenados en una tabla. En el siguiente ejemplo, se eliminan las tablas `Escribe`, `Autor` y `Libro`:

```
drop table Escribe;
drop table Autor;
drop table Libro;
```

6.3.2. SQL como Lenguaje de Manipulación de Datos (LMD)

En esta sección se describirán las instrucciones típicas más importantes de SQL para el soporte a las operaciones **CRUD** anteriormente introducidas. Existen, por tanto, cuatro tipos de sentencia para manipular los datos:

- ***Create***: implementada mediante la instrucción `insert`, sirve para insertar registros (también llamados **tuplas**) en una base de datos. En los ejemplos siguientes se insertan diversas tuplas en varias tablas, siguiendo el mismo orden en el que se especificaron los atributos cuando se creó la tabla. Así por ejemplo, se crea el código "*dbrown*" (como valor para el atributo `CodAutor`) para "*Dan Brown*", y el código "*cdv*" (como valor para el atributo `CodLibro`) para su libro "*El Código da Vinci*". El siguiente código SQL refleja las instrucciones necesarias:

```
 insert into Autor values ('dbrown', 'Dan', 'Brown', '', '',
↪ 'EstadoUnidense');
 insert into Libro values ('cdv', 'El Código da Vinci', 'Random House',
↪ '2003-04-23');
 insert into Escribe values ('dbrown', 'cdv');
```

- ***Read***, implementada mediante la instrucción `select`, permite hacer consultas a la base de datos. En los siguientes ejemplos se escribe el código que selecciona (1) el nombre y primer apellido de los autores con nacionalidad española ordenados por orden alfabético del Apellido1 y (2) se lista todos los libros que haya escrito el autor "*Pérez Reverte*" (cuyo `CodAutor` es "*perezreverte*").

```
Select Nombre, Apellido1 from Autor where Nacionalidad like 'Español' order
↪ by Apellido1;

Select Libro.Título from Autor, Escribe, Libro where ( Libro.CodLibro =
↪ Escribe.CodLibro and Autor.CodAutor = Escribe.CodAutor) and
↪ (Escribe.CodAutor ='perezreverte');
```

- ***Update*** permite actualizar los valores de las tuplas seleccionadas. En el siguiente ejemplo, se actualiza el valor del atributo `pseudonimo` al valor "*El Manco de Lepanto*" para el autor "*Miguel de Cervantes*" con `CodAutor` "*mcervantes*":

```
update Autor set pseudonimo="El Manco de Lepanto" where
↪ CodAutor='mcervantes';
```

- ***Delete*** cuyo principal objetivo es borrar o eliminar en bloque o de forma selectiva una o varias tuplas o registros de datos de una tabla relacional que cumplan una determinada condición. En el siguiente código SQL se borra la(s) tupla(s) que contiene(n) datos del autor "*Pérez Reverte*".

```
delete from Autor where CodAutor ='perezreverte';
```

6.3.3. SQL como Lenguaje de Administración de Datos (LAD)

Finalmente, SQL puede ser usado también para administrar los usuarios de una base de datos. Esto implica crear usuarios de la base de datos y asignarles diferentes tipos de permisos para realizar los diferentes tipos de operaciones vistos anteriormente sobre los distintos componentes de datos. Por ejemplo, se puede usar la siguiente instrucción para crear un usuario llamado `Ismael.Caballero` que tenga por contraseña `LibroMDSR`:

```
create user 'Ismael.Caballero' identified by 'LibroMDSR';
```

y la siguiente instrucción se usa para asignar al usuario `Ismael.Caballero` los permisos necesarios para el acceso, lectura, selección, inserción, actualización y borrado de los valores de la base de datos `Biblioteca` así como para poder modificar la estructura de los componentes de la base de datos `Biblioteca` creada anteriormente:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER ON biblioteca.*
↪ TO 'Ismael.Caballero'@'%';
FLUSH PRIVILEGES;
```

Recuérdese que los ejemplos mostrados han sido realizados para MySQL 8, aunque la sintaxis no debería ser muy diferente para ser usadas en otros SGBDR.

6.4. Usando bases de datos desde R

Como el propósito de este libro es aprender los fundamentos de la Ciencia de Datos usando el software R, en las siguientes secciones se explicarán cómo implementar las operaciones CRUD

usando sentencias de paquetes específicos de R. Como en este capítulo se ha usado MySQL, se ha usado el *driver* específico de `RMySQL`. En caso de que se usase otro sistema gestor de bases de datos se tendría que recurrir al paquete específico que contuviera el driver correspondiente. Por tanto, en las siguientes secciones se explicará cómo conectarse a una base de datos usando las funciones correspondientes y cómo se implementan las operaciones CRUD con funciones del paquete `RMySQL`.

6.4.1. Conexión a una base de datos

Antes de poder realizar ninguna operación con la base de datos gestionadas por MySQL es preciso instalar y cargar los paquetes necesarios:

```
install.packages("RMySQL")
library(RMySQL)

library(DBI)
```

Para desarrollar las explicaciones, se ha usado una base de datos llamada `classimodels` que ha sido implementada para MySQL v8.0 y desplegada en un servidor con dirección IP 172.20.48.118 que está escuchando en el puerto 3306. El usuario que se conectará a la base de datos será `Ismael.Caballero` siendo su contraseña `MdsR.2022`. Otros usuarios tendrían que modificar los parámetros correspondientes para realizar las conexiones a sus propias bases de datos. Se almacenarán todos estos datos en variables para hacer más sencillo el mantenimiento de los chunks. Con `dbConect()` se realiza la conexión. Con `summary()` o con `dbGetInfo()` se pueden mostrar los resultados de la conexión en caso de que ésta se haya realizado con éxito. Una vez terminada todas las tareas con la base de datos debería desconectarse mediante la instrucción `dbDisconnect()`. En esta chunk, se deja comentada para poder hacer el resto de los ejemplos del capítulo.

```
usuario <- "Ismael.Caballero"
passwd <- "MdsR.2022#"
nombrebd <- "classicmodels"

servidor <- "172.20.48.118"
puerto <- 3306

mibbdd <- dbConnect(MySQL(),
  user = usuario, password = passwd, dbname = nombrebd,
  host = servidor, port = puerto
)
summary(mibbdd)
dbGetInfo(mibbdd)
dbDisconnect(mibdd)
```

6.4.2. Operaciones de lectura / selección (*read*) de datos

Una de las operaciones más frecuentes en cualquier tipo de bases de datos son las de lecturas o consultas (*read*), implementadas en SQL con las sentencias de tipo `select`. El driver `RMySQL` ofrece distintas alternativas para realizar consultas de selección en **R Software**. La elección de la mejor operación dependerá de la complejidad de las consultas que se quiera realizar. Se listan a continuación:

1. `dbReadTable()`, que permite leer una tabla entera de una base de datos MySQL. Es recomendable usar este método si la tabla no es excesivamente grande. Se pueden almacenar los resultados en un data frame para hacer operaciones con ellos después. La ventaja es que sabiendo manejar data frames en R no se necesita aprender mucho más detalle del lenguaje SQL como LMD, pero la desventaja es que se acaba perdiendo parte del potencial expresivo de SQL para hacer algunas cosas más fáciles y eficientes. En el siguiente ejemplo se muestra cómo cargar toda la tabla `customers` en un data frame llamado `tblCustomers`, mostrando los resultados con la instrucción `summary()`:

```
tblCustomers <- dbReadTable(mibbdd, "customers")
# summary (tblCustomers)
```

2. `dbGetQuery()`, que tiene más flexibilidad que `dbReadTable()` porque permite mediante una sentencia SQL `select` (véase más información en el tutorial de SQL en W3C o en la página oficial de "select" sobre MySQL) particularizar la consulta a la base de datos. Esto puede implicar la selección de atributos específicos o incluso el uso de filtros sobre los atributos seleccionados. Por ejemplo, si se quisieran recuperar el número y el nombre de los clientes de Madrid, se podría personalizar la consulta añadiendo las condiciones correspondientes en la cláusula `where` como se muestra en siguiente código. Puede obtenerse más información sobre la cláusula where. Por comodidad se escribe aparte la consulta SQL en una variable para poder manejar más fácilmente la operativa en **R Software**. Escribir esta consulta puede ser lo que entrañe más dificultad. A continuación, se ejecuta la consulta con `dbGetQuery()` y se almacenan los resultados en un data frame para su uso posterior. Nuevamente se comprueba el resultado con la instrucción `sumary()`.

```
SentenciaSQL_NombresClientes <- "Select CustomerNumber, CustomerName from
↪  customers where city = 'Madrid'"

Consulta_ClientesMadrid <- dbGetQuery(mibbdd, SentenciaSQL_NombresClientes)

summary(Consulta_ClientesMadrid)
```

Teniendo los resultados en data frames, ya es posible procesarlos en **R** como si fuera cualquier otro tipo de datos.

Obsérvese que las instrucciones siguientes serían equivalentes:

```
dbReadTable(mibbdd, "customers")
dbGetQuery(mibbdd, "select * from customers")
```

3. `dbSendQuery()` combinado con `dbFetch()`. La principal diferencia entre `dbSendQuery()` y `dbGetQuery()` es que la primera no recupera datos de la base de datos, y hay que traerlos explícitamente con la función `dbFetch()`. En el siguiente fragmento de código se muestra la forma de utilización de ambas funciones con un resultado exactamente igual que en el apartado anterior el anterior. En cualquier caso, es interesante resaltar antes algunos aspectos interesantes:

GEMA!! ¿Habría alguna forma de que esto tuviera un nivel de indentación?

- La información de la consulta generada con `dbSendQuery()` puede ser mostrada usando la función `dbGetInfo()`.
- Es posible recordar la consulta SQL que se utilizó en `dbSendQuery()` mediante la función `dbGetStatement()`.
- La función `dbFetch()` tiene dos argumentos: la consulta y el número de registros a recuperar (que se puede consultar con la función); si se quieren recuperar todos los registros que haya podido producir la consulta, debe pasarse el argumento `n=-1`.
- Si se quiere saber el número de elementos que se han traído con la función `dbFetch()` se puede usar la función `dbGetRowCount()`

Esto siguiente también necesitaría indentación ..>

En el siguiente ejemplo se muestra cómo leer los datos de la tabla `customers` combinando `dbSendQuery()` con `dbFetch()`. La sentencia SQL es la misma pero se va a realizar la recuperación de los datos de forma diferente con la función `dbSendQuery()` (el resultado se almacena en el data frame `Consulta`). Con `dbFetch` se recuperan los resultados de las consulta desde la base de datos y se almacenan en la variable `Consulta_ClientesMadrid_condbSendQuery`. Con `dbGetInfo()` se muestran los metadatos de la consulta.

```
SentenciaSQL_NombresClientes <- "Select CustomerNumber, CustomerName from
↪  customers where city = 'Madrid'"

Consulta <- dbSendQuery(mibbdd, SentenciaSQL_NombresClientes)
dbGetInfo(Consulta)
#> $statement
#> [1] "Select CustomerNumber, CustomerName from customers where city =
↪  'Madrid'"
#>
#> $isSelect
#> [1] 1
#>
#> $rowsAffected
```

```
#> [1] -1
#>
#> $rowCount
#> [1] 0
#>
#> $completed
#> [1] 0
#>
#> $fieldDescription
#> $fieldDescription[[1]]
#> NULL

print(paste("Consulta realizada:", dbGetStatement(Consulta)))

Consulta_ClientesMadrid_condbSendQuery <- dbFetch(Consulta, n = -1)
print(paste("Número de elementos devueltos en la consulta",
↪  dbGetRowCount(Consulta)))

summary(Consulta_ClientesMadrid_condbSendQuery)
```

Para extraer información de los resultados de la consulta, se puede usar la función `dbColumnInfo()`:

```
dbColumnInfo(Consulta)
#> name Sclass type length
#> 1 CustomerNumber integer INTEGER 11
#> 2 CustomerName character VAR_STRING 200
```

El driver `RMySQL` no proporciona funciones para conocer directamente el tipo y tamaño de los atributos de una tabla. Obsérvese, que mediante la función `dbSendQuery()` y `dbColumnInfo()` se podría obtener esta información haciendo una consulta que incluyera los atributos en los cuales se está interesados. Por ejemplo, para conocer el tipo de datos y tamaño de los atributos de la tabla `employees` se podría usar el siguiente fragmento de código. Con `dbColumnInfo()` se muestran los metadatos de implementación (operativos) de los atributos de la tabla `Employees`. Finalmente con la instrucción `dbClearResult(ConsultaEmployees)` se podrán limpiar los resutlados de la consulta para optimizar el sistema.

```
SentenciaSQL_TablaEmployees <- "Select * from employees"
ConsultaEmployees <- dbSendQuery(mibbdd, SentenciaSQL_TablaEmployees)

dbColumnInfo(ConsultaEmployees)

#> name Sclass type length
#> 1 employeeNumber integer INTEGER 11
```

```
#> 2 lastName character VAR_STRING 200
#> 3 firstName character VAR_STRING 200
#> 4 extension character VAR_STRING 40
#> 5 email character VAR_STRING 400
#> 6 officeCode character VAR_STRING 40
#> 7 reportsTo integer INTEGER 11
#> 8 jobTitle character VAR_STRING 200

dbClearResult(ConsultaEmployees)
#> [1] TRUE
```

6.4.3. Operaciones de inserción (*create*) y actualización (*update*) de datos

Antes de almacenar los datos en la base de datos, es necesario crear las estructuras necesarias, que como se ha venido diciendo serán tablas y atributos. Para ello se utilizarán las instrucciones especiales de SQL como lenguaje de definición de datos (LDD), esto incluye instrucciones para crear tablas (`create table`), para modificar tabla (`alter table`) o para borrar una tabla (`drop table`).

Para poder hacer operaciones con los datos, es preciso crear usuarios y asignarles los privilegios adecuados sobre las tablas y atributos. Esto se consigue utilizando las instrucciones especiales de SQL como lenguaje de administración de datos (LAD), lo que incluye instrucciones para crear usuarios (`create user`), modificar ciertos aspectos de los usuarios (`alter user`), y para borrar usuarios (`drop user`).

Entonces, un usuario de la base de datos que tenga privilegios suficientes sobre las estructuras creadas puede crear (`insert`) o modificar (`update`) registros de datos usando las instrucciones específicas de SQL como lenguaje de manipulación de datos (LMD)-véase cómo otorgar privilegios a un usuario para crear tablas:

No obstante, y dado que el objetivo del libro es R, se deja fuera del alcance de este capítulo el uso de los aspectos DDL, DML y DAL de SQL, y se cubrirán mediante la instrucciones `dbWriteTable()` de `RMySQL` los aspectos de inserción y de actualización de los registros. `dbWriteTable()` se usa por tanto para exportar datos de R a una base de datos MySQL, y puede ser usado para las siguientes acciones, siempre y cuando el usuario que ejecute las acciones tenga suficientes permisos para realizarlas.

6.4.3.1. Crear una nueva tabla con datos

Se describe cómo crear una tabla nueva a partir de un data frame que se puebla con datos iniciales. Es importante observar que el data frame tendrá tantas columnas como atributos se pretenda o se necesita que tenga la tabla. Por ejemplo, se va a crear un data frame llamado `DatosPrueba` con dos columnas, una de tipo numérico llamada `CodPrueba`, y otra de tipo texto llamada `DatosPrueba`. En este ejemplo y a modo ilustrativo, los datos serán completamente

aleatorios, pero en un caso real, el usuario deberá utilizar los valores adecuados. A continuación se construye el dataframe `dfDatosPruebas` y mediante `dbListTables()` se comprueba que la tabla no existe en la conexión a la base de datos. Finalmente con `dbWriteTable()` se creará la nueva tabla. Es importante tener en cuenta las posibles conexiones simultáneas a la base de datos porque se podrían generar problemas. Finalmente, con `dbWriteTable()` se podría comprobar si se ha creado correctamente otra tabla.

```
CodPrueba <- c(1:26)
NombrePrueba <- c(letters[1:26])

dfDatosPrueba <- data.frame(CodPrueba, NombrePrueba)

dbListTables(mibbdd)
#> [1] "Autor" "DatosPrueba_16" "DatosPrueba_22" "Datos_Prueba_01",
#> [5] "Make" "Pelicula" "customers" "employees",
#> [9] "offices" "orderdetails" "orders""payments",
#> [13] "productlines" "products"

dbWriteTable(mibbdd, "DatosPrueba", dfDatosPrueba, overwrite = TRUE, row.names
↪  = FALSE)
#> [1] TRUE
#>
dbListTables(mibbdd)
#> [1] "Autor" "DatosPrueba" "DatosPrueba_16" "DatosPrueba_22",
#> [5] "Datos_Prueba_01" "Make" "Pelicula" "customers" ,
#> [9] "employees" "offices" "orderdetails","orders" ,
#> [13] "payments" "productlines" "products"
```

Es interesante pensar en la utilidad de este método para duplicar tablas.

6.4.3.2. Sobreescribir una tabla existente con datos actualizados

Cuando se trata de actualizar algunos valores de los atributos de la tabla o de añadir nuevos registros a la tabla, la operación será básicamente la misma que antes, pero se tendrá primero que leer la tabla y convertirla en un data frame para actualizar los valores correspondientemente en el data frame con los valores adecuados o añadir los nuevos valores (en este caso añadimos una nueva fila) y finalmente volver a utilizar el comando `dbWriteTable()` añadiendo el parámetro `overwrite = TRUE` para sobrescribir la tabla y asegurándonos que se añade el parámetro `row.names =FALSE`. En el siguiente ejemplo se actualizan los valores de una tupla específica.

```
dfDatosPrueba <- dbReadTable(mibbdd, "DatosPrueba")

dfDatosPrueba$NombrePrueba[25] <- "en un lugar de la mancha"
```

```
dbWriteTable(mibbdd, "DatosPrueba", dfDatosPrueba, overwrite = TRUE, row.names
↪  = FALSE)
#> [1] TRUE

dfDatosPruebaModificado <- dbReadTable(mibbdd, "DatosPrueba")
```

6.4.3.3. Añadir nuevos registros a una tabla

Existen dos estrategias para añadir registros a una tabla.

La primera es utilizando la técnica de la sobreescritura descrita anteriormente. Para ello, se procede como antes: se carga la tabla en un data frame (en este caso `dfDatosPrueba`), se añaden nuevas filas (registros) al data frame (cargadas previamente en el data frame `dfNuevoRegistro`) usando `rbind()`, y a continuación se sobreescribe la tabla usando `dbWriteTable()`. En este fragmento de código se muestra cómo se implementa esta estrategia. En el siguiente ejemplo se muestra cómo añadir nuevos registros a una tabla sobreescribiéndola completamente.

```
dfDatosPrueba <- dbReadTable(mibbdd, "DatosPrueba")

dfNuevoRegistro <- as.list(dfDatosPrueba)

dfNuevoRegistro$CodPrueba <- c(27)
dfNuevoRegistro$NombrePrueba <- c("Un Valor Nuevo")

dfDatosPrueba <- rbind(dfDatosPrueba, dfNuevoRegistro)

dbWriteTable(mibbdd, "DatosPrueba", dfDatosPrueba, overwrite = TRUE, row.names
↪  = FALSE)
#> [1] TRUE
```

La opción anterior puede ser interesante si la tabla no tiene muchos registros y el coste computacional no es muy grande. Pero si se tienen muchos registros es preferible usar otra estrategia para añadir un nuevo registro a la tabla. En esta ocasión, se puede hacer creando un data frame compatible con la estructura de la tabla, y ejecutar la instrucción `dbWriteTable()` asegurando que se pone el parámetro `append = TRUE`. Esto añadirá el nuevo registro al final de la tabla. El siguiente fragmento de código muestra cómo realizar esta operación.

```
dfDatosPruebaNuevos <- as.list(dfDatosPrueba)

dfDatosPruebaNuevos$CodPrueba <- 28
dfDatosPruebaNuevos$NombrePrueba <- "Otro valor nuevo"

dfDatosPruebaNuevos <- data.frame(dfDatosPruebaNuevos)
```

```
dbWriteTable(mibbdd, "DatosPrueba", dfDatosPruebaNuevos, append = TRUE,
↪  row.names = FALSE)
#> [1] TRUE
```

6.4.3.4. Inserción con consulta SQL usando la instrucción `dbSendQuery()`

Una última forma de insertar valores en una tabla es mediante la instrucción `dbSendQuery` utilizando una consulta de inserción `insert`. En el siguiente ejemplo se muestra cómo insertar tuplas o registros mediante `dbSendQuery()`, en este caso, se añaden datos completamente aleatorios.

```
SentenciaSQL_Insercion <- "insert into DatosPrueba value (29, 'Una tercera
↪  forma')"

dbSendQuery(mibbdd, SentenciaSQL_Insercion)
#> <MySQLResult:-365007472,0,23>
```

6.4.4. Operaciones de Borrado de datos (*delete*)

Finalmente, se describen las operaciones de borrado. Análogamente a como se hacían las operaciones de inserción, se pueden hacer de dos formas:

1. **Borrado de valores usando `dbWriteTable()` con sobreescritura**: Esto implica extraer todos los datos de la tabla, borra el registro o los registros correspondientes y luego sobreescribir nuevamente la tabla en la base de datos mediante la instrucción `dbWriteTable()` con la opción `overwrite = TRUE`. El siguiente fragmento de código muestra cómo hacerlo, y finalmente para ver los resultados se pueden usar la función `summary()`.

```
dfDatosPrueba <- dbReadTable(mibbdd, "DatosPrueba")

dfDatosPrueba <- dfDatosPrueba[dfDatosPrueba$CodPrueba < 25, ]

dfNuevoRegistro$CodPrueba <- c(101)
dfNuevoRegistro$NombrePrueba <- c("Un Valor Nuevo")

dfDatosPrueba <- rbind(dfDatosPrueba, dfNuevoRegistro)

dbWriteTable(mibbdd, "DatosPrueba", dfDatosPrueba, overwrite = TRUE, row.names
↪  = FALSE)
#> [1] TRUE
```

2. **Borrado de registros con consulta SQL en `dbSendQuery()`**: Es posible utilizar una sentencia SQL de borrado delete con la instrucción `dbSendQuery()` para borrar registros de la base de datos. El siguiente fragmento de código muestra cómo hacerlo.

```
# Se usa una sentencia SQL de borrado. El criterio de borrado es completamente
↪  aleatorio a efectos ilustrativos.
SentenciaSQL_Eliminación <- "delete from DatosPrueba where CodPrueba > 10"
dbSendQuery(mibbdd, SentenciaSQL_Eliminación)
#> <MySQLResult:1,0,30>
```

6.4.4.1. Borrado de toda la tabla con consulta SQL en `dbSendQuery`

Finalmente si fuera necesario eliminar toda la tabla se podría usar una sentencia drop table

```
dbSendQuery(mibbdd, "drop table DatosPrueba")
#> <MySQLResult:0,0,31>
dbDisconnect(mibbdd)
#> [1] TRUE
```

Resumen
En este capítulo se ha presentado los fundamentos de las bases de datos relacionales. Es importante tener presente los siguientes aspectos:

- Los datos en las bases de datos se corresponden a valores de atributos relevantes de entidades del mundo real.
- Los datos de una base de datos son una percepción u observación del mundo real.
- Los datos son las materias primas de los procesos de negocio.
- Los sistemas de Información dan soporte a los procesos de negocio.
- Los datos son un elemento fundamental de los sistemas de información.
- Existen diferentes vistas o percepciones de los datos que se van refinando a medida que se implementan las bases de datos. Estas vistas se representan mediante diferentes modelos: conceptual, lógico y físico.
- Cada uno de los modelos anteriores utiliza lenguajes que tienen capacidades descriptivas (y por tanto, limitaciones diferentes).
- SQL (**Structured Query Language**) es el lenguaje más comúnmente utilizado en operaciones sobre el modelo físico de bases de datos relacionales.
- SQL se puede utilizar como Lenguaje de Definición de Datos (LDD), como Lenguaje de Manipulación de Datos (LMD), y como Lenguaje de Administración de Datos (LAD).
- La sintaxis de SQL depende fuertemente del Sistema Gestor de Bases de Datos Relacionales que lo implemente.
- **R Software**, a través del driver específico, permite manejar bases de datos implementando las operaciones CRUD.

Capítulo 7

Gesitón de bases de datos NoSQL

Ricardo Pérez-Castillo e Ismael Caballero

7.1. Introducción al Big Data

Para pensar en Big Data, se tiene que saber que actualmente se vive en la era de la información, con un teléfono móvil en cada bolsillo, un ordenador portátil en cada mochila y grandes sistemas de tecnología funcionando diariamente mandando datos y datos cada segundo. Se ve claramente que el mundo tiene más datos que nunca, pero esto no es todo, ya que día a día crece aún más (López, 2012).

En particular, el volumen de datos disponibles para las empresas aumentó drásticamente desde 2004. En 2004, la cantidad total de datos almacenados en Internet fue de 1 petabyte (equivalente a 100 años de todo el contenido de televisión). Para 2011 la cantidad total de información almacenada en todo el mundo fue de 1 zettabyte (1 millón de petabytes o 36 millones de años de video de alta definición [HD]). Para el año 2015, alcanzó 7.9 zettabytes (o 7.9 millones de petabytes) y para 2020 se disparó a 35 zettabytes (o 35 millones de petabytes). Este gran volumen de datos y su crecimiento continuo y exponencial, supera las capacidades de las herramientas de datos tradicionales para capturar, almacenar, administrar y analizar los datos (Kalyvas and Overly, 2014). Por este motivo se hace necesario el uso de métodos, técnicas y herramientas de gestión de datos nuevas. Este espacio es el que cubre Big Data.

Big data es un término abstracto que en cierta medida se ha puesto de moda en diferentes ámbitos: desde los negocios, marketing, social media, y en diferentes ingenierías como informática, sistemas de información, almacenamiento y recuperación de datos, etc. Big data es un término que hace referencia al gran volumen de datos (tanto estructurados como no estructurados) y que inundan los negocios día a día. Pero no es la cantidad de datos lo más relevante. Lo que realmente importa es lo que las organizaciones hacen con los datos. Los grandes volúmenes de datos grandes se pueden analizar en busca de ideas que conducen a mejores decisiones y movimientos comerciales estratégicos (SAS Institute Inc., 2017).

Cuando se acumulan grandes volúmenes de datos se plantea la necesidad de ver qué se puede hacer con ellos. Esto redunda en la necesidad de gestionar los datos con una finalidad organizativa, a disponer de tecnología y metodologías específicas. La misma gestión de datos nos lleva a generar información que sea relevante en el contexto de la organización, es decir, a generar conocimiento para la acción, y que sea aplicable por ejemplo a la toma de decisiones, al diseño de acciones para la organización o a la elaboración de planes estratégicos (García-Alsina, 2017).

Por lo tanto, cuando se habla de datos masivos, se está hablando también de gestión de la información y de generación de conocimiento para la acción. Este campo científico es el que da las pautas metodológicas para gestionar grandes volúmenes de datos con el fin de crear valor mediante una serie de procesos y procedimientos. Pero, además, hemos de tener en cuenta que debemos contar con la tecnología, para capturar datos, procesarlos, analizarlos e interpretarlos de manera rápida y eficiente (García-Alsina, 2017, Gómez García and Conesa i Caralt (2015)).

Al final, del conjunto de definiciones previas, se puede concluir que Big Data es el ***"conjunto de datos masivos heterogéneos que supera la capacidad del software habitual para ser capturados, gestionados y procesados en un tiempo razonable"***. Esta definición tiene en cuenta tres de las V's del Big Data (presentada en la sección 7.2): volumen, variabilidad y velocidad. Así, cuando se habla de Big Data nos referimos a conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño (Volumen), complejidad (Variabilidad) y velocidad de crecimiento (Velocidad) dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales, tales como bases de datos relacionales y estadísticas convencionales o paquetes de visualización, dentro del tiempo necesario para que sean útiles.

7.2. Las V's del Big Data

Originalmente se han definido las tres **V's** del Big Data como: *Volumen*, *Variedad* y *Velocidad*. Los cuales son tres aspectos clave que deben considerar los entornos Big Data.

El **volumen** se refiere a la cantidad de datos que son generados cada segundo, minuto y día en nuestro entorno. Es la característica más asociada al Big Data, ya que hace referencia a las cantidades masivas de datos que se almacenan con la finalidad de procesar dicha información, transformando los datos en acciones. La importancia del volumen de datos ya se trató en las secciones anteriores. Cada vez las personas están más conectadas al mundo digital por lo que se genera más y más datos. Para algunas empresas, el estar en el mundo digital es algo obligatorio, por lo que la cantidad de datos generados es aún mayor. Por ejemplo, una empresa que vende sus productos únicamente a través de un canal online, le convendría implantar tecnología Big Data para procesar toda aquella información que recoge su página web rastreando todas las acciones que lleva a cabo el cliente; conocer donde cliquea más veces, cuántas veces ha pasado por el carrito de la compra, cuáles son los productos más vistos, las páginas más visitadas, etc.

La **velocidad** se refiere a los datos en movimiento por las constantes interconexiones que realizamos, es decir, a la rapidez en la que son creados, almacenados y procesados en tiempo real. Para los procesos en los que el tiempo resulta fundamental, tales como la detección de fraude en una transacción bancaria o la monitorización de un evento en redes sociales, estos tipos de datos deben estudiarse en tiempo real para que resulten útiles para el negocio y se consigan conclusiones efectivas. El procesamiento de los datos masivos se debe hacer en el mínimo tiempo

posible, y en algunas ocasiones en tiempo real. Es un tema especialmente delicado cuando se deben tomar decisiones en contextos críticos; como por ejemplo, en catástrofes naturales o pandemias, o cuando se efectúa el seguimiento de una campaña analizando comentarios de los actores a quienes esta va dirigida, para ir reorientándola en función de la retroalimentaciones que fluye en redes sociales (García-Alsina, 2017).

La **variedad** se refiere a las formas, tipos y fuentes en las que se registran los datos. Estos datos pueden ser datos estructurados y fáciles de gestionar como son las bases de datos, o datos no estructurados, entre los que se incluyen documentos de texto, correos electrónicos, datos de sensores, audios, vídeos o imágenes que tenemos en nuestro dispositivo móvil, hasta publicaciones en nuestros perfiles de redes sociales, artículos que leemos en blogs, las secuencias de click que hacemos en una misma página, formularios de registro e infinidad de acciones más que realizamos desde nuestro Smartphone, Tablet y ordenador. Estos últimos datos requieren de una herramienta específica, debido a que el tratamiento de la información es totalmente diferente con respecto a los datos estructurados. Para ello, las empresas necesitan integrar, observar y procesar datos que son recogidos a través de múltiples fuentes de información con herramientas apropiadas.

Después, otros autores han ido incorporando progresivamente otras V's que añaden otros aspectos clave para ser considerados. De esta forma, se se habla de V's en Big Data no se puedepasar por alto la principal característica del análisis de datos que es la V de **valor** de los datos. Así pues, en los últimos trabajos se habla de las 7 V's del Big Data (IIC, 2016): **veracidad**, **viabilidad**, **visualización**, y **valor** de los datos.

7.3. Fuentes de Datos en entornos Big Data

En esta sección se muestran los diferentes tipos de datos y flujos de datos, como son: estructurados, no estructurados o semiestructurados.

- **Datos estructurados**: Son aquellos datos que tienen Longitud y formato: como las fechas, los números o las cadenas de caracteres. En esta categoría entran los que se compilan en los censos de población, los diferentes tipos de encuestas, los datos de transacciones bancarias, las compras en tiendas online, etc.
- **Datos no estructurados**: Son los datos que carecen de un formato determinado y no pueden ser almacenados en una tabla. Pueden ser de tipo texto (los que generan los usuarios de foros, redes sociales, documentos de Word), y los de tipo no-texto (cualquier fichero de imagen, audio, vídeo). Dentro de esta categoría están: archivos de audio, video, fotografías, texto, correos electrónicos, etc. Conviene saber que este tipo de datos no tiene campos fijos y normalmente se tiene poco control sobre ellos. Su manipulación requiere herramientas como Hadoop (la más popular) y/o bases de datos ***NoSQL***, entre otras.
- **Datos semiestructurados**: Poseen organización interna o marcadores que facilita el tratamiento de sus elementos. No pertenecen a bases de datos relacionales. En este caso estaríamos hablando de documentos XML, HTML o los datos almacenados en bases de datos ***NoSQL***. Los cuales tienen una cierta estructura, aunque sin llegar a estar totalmente estructurados. Además, de acuerdo con otras fuentes, también podemos considerar

en esta clasificación datos Multi-estructurados o híbridos, entre los que pondríamos datos de mercados emergentes, *e-commerce*, datos meteorológicos.

7.4. Bases de datos Relacionales vs. NoSQL

Lo que hace que Big Data sea tan útil para muchas empresas es el hecho de que proporciona respuestas a muchas preguntas que las empresas ni siquiera sabían que tenían. En otras palabras, proporciona un punto de referencia. Con una cantidad tan grande de información, los datos pueden ser moldeados o probados de cualquier manera que la empresa considere adecuada. Al hacerlo, las organizaciones son capaces de identificar los problemas de una forma más comprensible.

La recopilación de grandes cantidades de datos y la búsqueda de tendencias dentro de los datos permiten que las empresas se muevan mucho más rápidamente, sin problemas y de manera eficiente. También les permite eliminar las áreas problemáticas antes de que los problemas acaben con sus beneficios o su reputación.

El análisis de Big Data ayuda a las organizaciones a aprovechar sus datos y utilizarlos para identificar nuevas oportunidades. Eso, a su vez, conduce a movimientos de negocios más inteligentes, operaciones más eficientes, mayores ganancias y clientes más satisfechos Las empresas con más éxito con Big Data consiguen valor de las siguientes formas:

- **Reducción de coste.** Las grandes tecnologías de datos, como Hadoop y el análisis basado en la nube, aportan importantes ventajas en términos de costes cuando se trata de almacenar grandes cantidades de datos, además de identificar maneras más eficientes de hacer negocios.
- **Más rápida y mejor toma de decisiones.** Con la velocidad de Hadoop y la analítica en memoria, combinada con la capacidad de analizar nuevas fuentes de datos, las empresas pueden analizar la información inmediatamente y tomar decisiones basadas en lo que han aprendido.
- **Nuevos productos y servicios.** Con la capacidad de medir las necesidades de los clientes y la satisfacción a través de análisis viene el poder de dar a los clientes lo que quieren. Con la analítica de Big Data, más empresas están creando nuevos productos para satisfacer las necesidades de los clientes.

Existen ciertas diferencias entre las fuentes de datos tradicionales y las nuevas fuentes de datos que considera Big Data como se resumen en la Tabla 7.1.

Tabla 7.1: . Diferencias entre tecnologías tradicionales y tecnologías Big Data.

Tecnologías Tradicionales	Tecnologías Big Data
Bases de datos Relacionales	Bases de datos Relacionales + *NoSQL*
Consultas	Consultas, Caputras y Procesamientos

Tecnologías Tradicionales	Tecnologías Big Data
Datos Internos	Datos Heterogéneos
Ámbito de la Informática	Todos los ámbitos

En primer lugar, la tecnología tradicional de almacenamiento y gestión de datos (desde final de los años 80) ha sido las bases de datos relacionales. Aunque las bases de datos relacionales no son ni mucho menos una tecnología en desuso, los entornos Big Data consideran otras tecnologías como por ejemplo las bases de datos ***NoSQL***, las cuales son bases de datos no relacionales optimizadas para modelos de datos sin esquema y de desempeño escalable. También son ampliamente conocidas por su facilidad de desarrollo, baja latencia y resiliencia. Utilizan una variedad de modelos de datos, como los almacenes de valor clave en memoria, de gráficos, de documentos y en columnas. Esta página incluye recursos que le servirán para comenzar a usar las bases de datos ***NoSQL*** (Amazon Web Services, 2018).

Otra gran diferencia respecto a las tecnologías tradicionales es que los entornos Big Data, no sólo se centran en la consulta de datos, sino también en su captura y procesamiento (véase Tabla 1). Además, las fuentes de datos pueden proveer datos heterogéneos con formatos heterogéneos. Estas diferencias hacen que un ámbito puramente informático no sea necesario en los entornos Big Data.

7.5. Bases de datos NoSQL

7.5.1. Definición de bases de datos NoSQL

El término ***NoSQL*** se usa la primera vez en 1998 para referirse a una base de datos relacional sin SQL (Strozzi, 1998). ***NoSQL*** no significa estar en contra de SQL, y de hecho esto suele ser una falacia encontrada en la literatura. Sin embargo, para determinados problemas hay otras soluciones de almacenamiento más apropiadas.

Hay una gran variedad de sistemas de gestión de bases de datos que no usan SQL como el principal lenguaje de consultas. Los datos almacenados no requieren estructuras fijas como tablas y no se garantiza completamente los *principios ACID* que si deben cumplir las bases de datos relacionales (SQL):

- **Atomic** (Atomicidad). Las transacciones se ejecutan completamente o no. Si fallan es como si ni siquiera se hubieran intentado ejecutar.
- **Consistency** (Consistencia). Un sistema consistente garantiza cualquier transacción llevará a la base de datos de un estado válido a otro estado válido. Cualquier dato que se escriba en la base de datos tiene que ser válido de acuerdo con todas las reglas de integridad definidas en el modelo.
- **Isolation** (Aislamiento). Cuando varias transacciones se ejecutan en paralelo, cada una de ellas ve el sistema de la misma manera a como lo vería si se ejecutaran de forma aislada o secuencial.

- **Durability** (Durabilidad). El resultado de las transacciones es un cambio en el estado del sistema persistente. Si apagamos la máquina y la arrancamos de nuevo el cambio producido por la transacción aún está presente.

Los sistemas ***NoSQL*** se denominan a veces *"no sólo SQL"* para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

Se puede decir que la aparición del término ***NoSQL*** aparece con la llegada de la web 2.0 ya que hasta ese momento sólo subían contenido a la red aquellas empresas que tenían un portal, pero con la llegada de aplicaciones como *Facebook*, *Twitter* o *Youtube*, entre otras, cualquier usuario podía subir contenido, provocando así un crecimiento exponencial de los datos (acens.com, 2014).

Es en este momento cuando empiezan a aparecer los primeros problemas de la gestión de toda esa información almacenada en bases de datos relacionales. En un principio, para solucionar estos problemas de accesibilidad, las empresas optaron por utilizar un mayor número de máquinas, pero pronto se dieron cuenta de que esto no solucionaba el problema, además de ser una solución muy cara. La otra solución era la creación de sistemas pensados para un uso específico que con el paso del tiempo han dado lugar a soluciones robustas, apareciendo así el movimiento ***NoSQL*** (acens.com, 2014).

Por lo tanto, hablar de bases de datos ***NoSQL*** es hablar de estructuras que permiten almacenar información en aquellas situaciones en las que las bases de datos relacionales generan ciertos problemas debido principalmente a problemas de escalabilidad y rendimiento de las bases de datos relacionales donde se dan cita miles de usuarios concurrentes y con millones de consultas diarias.

Además de lo comentado anteriormente, las bases de datos ***NoSQL*** son sistemas de almacenamiento de información que no cumplen con el esquema entidad–relación. Tampoco utilizan una estructura de datos en forma de tabla donde se van almacenando los datos, sino que para el almacenamiento hacen uso de otros formatos como clave–valor, mapeo de columnas o grafos.

Las bases de datos relacionales modernas típicamente han mostrado poca eficiencia en determinadas aplicaciones que usan los datos de forma intensiva, incluyendo el indexado de un gran número de documentos, la presentación de páginas en sitios que tienen gran tráfico, y en sitios de streaming audiovisual. Las implementaciones típicas de los (SGBDR) se han afinado o bien para una cantidad pequeña pero frecuente de lecturas y escrituras o para un gran conjunto de transacciones que tiene pocos accesos de escritura. Por otra parte, ***NoSQL*** puede servir gran cantidad de carga de lecturas y escrituras.

7.5.2. Necesidades no cubiertas por las bases de datos relacionales

Las bases de datos no relacionales, también conocidas como bases de datos NoSQL, son principalmente bases de datos distribuidas de código abierto y escalables horizontalmente. Además, son bases de datos que no usan esquemas, por lo que ofrecen una fácil replicación y API sencilla para el acceso a un gran volumen de datos.

La necesidad de este tipo de bases de datos surge porque hay fuentes de datos que son difíciles de modelar en bases de datos relacionales. Algunos ejemplos son texto (datos no estructurados), procesado en *streaming* (flujo continuo de datos), bases de datos científicas (estructuras multidimensionales).

Además de estos tipos de fuentes de datos, las características de las nuevas aplicaciones de internet tales como redes sociales, juegos online, etc., hacen que las bases de datos no relacionales sean necesarias para conseguir mayor velocidad, escalabilidad, independencia de la localización, disponibilidad y mejor gestión de todos los tipos de datos.

- **Velocidad**. Para demostrar la importancia de la velocidad en internet, sirva como ejemplo como dos grandes compañías monetizan la velocidad de acceso: Amazon tiene estudiado que cuando el tiempo de respuesta disminuye 100ms los ingresos se compensan en un 1 %. Por otra parte, Yahoo asegura que el tráfico aumenta en un 9 % cuando el rendimiento mejora en 400ms. De ahí la importancia de este factor para la evolución hacia bases de datos NoSQL, y de hecho este factor entraña una de las conocidas **V's** del Big Data.
- **Escalabilidad**. Al principio la web se consideró una interfaz más, pero no es sólo eso; se ha convertido en un elemento generador y consumidor de datos (fundamentalmente semiestructurados y no estructurados). En el contexto actual, las compañías necesitan mantener la respuesta rápida, aunque se incremente cualquiera de los siguientes elementos: (i) el número de usuarios simultáneos; (ii) el volumen de datos; (iii) se ejecutan en clústeres de máquinas baratas. Además, la arquitectura de los sistemas NoSQL permite: (i) escalar sin disminuir el rendimiento; (ii) añadir nodos sobre la marcha sin interrupciones del servicio; (iii) no generan cuellos de botella. La Figura 7.1 muestra una representación comparativa de la escalabilidad de bases de datos NoSQL frente a relacionales. Como se observa, aunque las bases de datos tienen un mejor desempeño para volúmenes de datos reducidos, este se reduce drásticamente para grandes volúmenes de datos. Mientras, el rendimiento de las bases de datos NoSQL tiende a ser constante, por lo que escalan mejor para datos masivos.

- **Independencia de la localización**. La globalización del mercado en *World Wide Web* obliga a dar servicio rápido y en todas partes del mundo. Las bases de datos no relacionales son distribuidas de acuerdo con diferentes arquitecturas como "principal-secundario", o *"peer-to-peer"*.
- **Disponibilidad**. Similar a la independencia de la localización, la disponibilidad en el mercado WWW es uno de los factores más críticos, ya que se espera una disponibilidad de servicio 24x7. Es decir, hay que pasar de una alta disponibilidad a la disponibilidad continua, cuyas características son: (i) diseño sin principal-secundario; (ii) *Multi-data center*; (iii) disponibilidad *cloud*; (iv) Copias de datos y funcionalidad en múltiples localizaciones.
- **Gestión de todos los tipos de datos**. Un factor clave en las bases de datos no relacionales es la necesidad de manejar nuevos tipos de datos, tanto estructurados, como no estructurados y semiestructurados; y todo esto sin perder el enfoque de un almacenamiento eficiente. En ese sentido, a menudo, las bases de datos ***NoSQL*** están altamente optimizadas para las operaciones recuperar y agregar, y normalmente no ofrecen mucho más que la funcionalidad de almacenar los registros (p.ej. almacenamiento clave-valor). La

Figura 7.1: Comparativa de escalabilidad en bases de datos relacionales y NoSQL (adaptada de [@LO2017]).

pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos.

7.5.3. Tipos de almacenamiento en bases de datos NoSQL

Se pueden distinguir al menos 4 tipos de bases de datos ***NoSQL*** (Hecht and Jablonski, 2011): clave-valor, documental, en grafo, orientadas a columnas.

- **Almacenamiento clave-valor**. Los datos se almacenan de forma similar a los mapas o diccionarios de datos, donde se accede al dato a partir de una clave única. Los valores (datos) son aislados e independientes entre ellos, y no son interpretados por el sistema. Pueden ser variables simples como enteros o caracteres, u objetos. Por otro lado, este sistema de almacenamiento carece de una estructura de datos clara y establecida, por lo que no requiere un formateo de los datos muy estricto. Son útiles para operaciones simples basadas en las claves. *Apache Cassandra* es la tecnología de almacenamiento clave-valor más reconocida por los usuarios.
- **Almacenamiento documental**. Este tipo de base de datos almacena datos semi-estructurados. Los datos se llaman documentos, y pueden estar formateados en XML, JSON, Binary JSON o el que acepte la misma base de datos, pero suele ser un formato de texto. Un ejemplo de cómo se usa lo encontramos en un blog: se almacena el autor, la fecha, el título, el resumen y el contenido del post. Todos los documentos tienen una clave única con la que pueden ser accedidos e identificados explícitamente. Estos documentos no son opacos al sistema, por lo que pueden ser interpretados y lanzar consultas sobre ellos (véase la Fig. 7.2). *CouchDB* o *MongoDB* son quizá las más conocidas. Hay que hacer mención especial a *MapReduce*, una tecnología de *Google* inicialmente diseñada para su algoritmo PageRank, que permite seleccionar un subconjunto de datos, agruparlos o

reducirlos y cargarlos en otra colección, y a Hadoop que es una tecnología de *Apache* diseñada para almacenar y procesar grandes cantidades de datos. Por ejemplo, *MongoDB* es una base de datos orientada a documentos. Los documentos se guardan en BSON, que es una forma de representar de forma binaria objetos JSON. De esta forma, con el comando insert y pasando un objeto JSON, MongoDB crea automáticamente un documento y lo añade en la base de datos generando un ObjectId para el nuevo documento (Rubenfa, 2014). Este objeto está especialmente pensado para garantizar unicidad en entornos distribuidos como MongoDB. El campo está compuesto por 12 bytes. Los cuatro primeros bytes son un *timestamp* con los segundos desde el *epoch* de Unix; los tres siguientes bytes representan el identificador único de la máquina; los dos siguientes el identificador del proceso; y para finalizar los últimos tres bytes, son un campo incremental. En definitiva, los nueve primeros bytes nos garantizan un identificador único por segundo, máquina y proceso. Los tres últimos bytes, nos garantizan que cada segundo podemos insertar $2^{24} = 16.777.216$ documentos con un identificador distinto. Dado que el *ObjectId* está compuesto de esa manera, esto proporciona funcionalidades muy útiles. La primera es que nos puede dar una indicación del orden de creación de los documentos. También sirve para obtener la fecha de creación del documento.

Figura 7.2: Ejemplo representativo de base de datos NoSQL documental. Adaptado de [@Sanchez2017].

- **Almacenamiento en grafo**. Este tipo de almacenamiento maneja datos semi-estructurados y está basado en teoría de grafos. En estas bases de datos se establece que la información son los nodos y las relaciones entre la información son las aristas (algo similar al modelo relacional). Su mayor uso se contempla en casos de relacionar grandes cantidades de datos que pueden ser muy variables. Por ejemplo, los nodos pueden contener objetos, variables y atributos diferentes en unos y otros. Las operaciones de consulta con *join* se sustituyen por recorridos a través del grafo, y se guarda una lista de adyacencias entre los nodos. Encontramos un ejemplo en las redes sociales: en *Facebook* cada nodo se considera un usuario, que puede tener aristas de amistad con otros usuarios, o aristas de publicación con nodos de contenidos. Soluciones como *Neo4J* y *GraphDB28* son las más conocidas dentro de las bases de datos en grafo.

- **Almacenamiento orientado a columnas**. El almacenamiento orientado a columnas es similar al documental. Su modelo de datos es definido como "un mapa de datos multidimensional poco denso, distribuido y persistente" (Hecht and Jablonski, 2011). Se orienta a almacenar datos con tendencia a escalar horizontalmente, por lo que permite guardar diferentes atributos y objetos bajo una misma clave. A diferencia del documental y el clave-valor, en este caso se pueden almacenar varios atributos y objetos, pero no serán interpretables directamente por el sistema. Permite agrupar columnas en familias y guardar la información cronológicamente, mejorando el rendimiento. Esta tecnología se acostumbra a usar en casos con 100 o más atributos por clave. Su precursor es BigTable de Google, pero han aparecido nuevas soluciones como *HBase* o *HyperTable*.

7.5.4. Limitaciones de las bases de datos NoSQL

A pesar de las ventajas de las bases de datos ***NoSQL***, estas tienen una serie de limitaciones, tanto técnicas como de carácter no tecnológico. En cuanto a limitaciones técnicas se encuentran: Cómo modelar correctamente los datos para maximizar las capacidades; nivel bajo de seguridad; no soporta transacciones; falta de madurez en *Business Intelligence*; problemas de compatibilidad ya resueltos en los modelos relacionales, entre otros.

Además existen otras limitaciones de carácter no tecnológico, como puede ser: falta de expertos; resistencia al cambio; disponibilidad del vendedor; el código abierto puede implicar problema de soporte para las empresas.

7.6. Ejemplo de integración de una base de datos NoSQL y análisis de datos en R

En esta sección se verá como R puede ser utilizado para conectarnos a una base de datos ***NoSQL***. En particular ***MongoDB***. En la sección 7.6.1 se presenta una introducción a *MongoDB*. En la sección 7.6.2 se explican los paquetes de **R** utilizados para acceder a *MongoDB*. La sección 7.6.3 explica como conectarse a una base de datos ***MongoDB*** remota. Las secciones 7.6.4 y 7.6.5 realizan consultas y análisis sobre una colección de viajes realizados por los usuarios de un servicio de bicicletas compartidas con sede en la ciudad de *Nueva York*.

7.6.1. Introducción a MongoDB

MongoDB (de la palabra en inglés *"humongous"* que significa *enorme*) es un sistema de base de datos ***NoSQL*** orientado a documentos, desarrollado bajo el concepto de código abierto. ***MongoDB*** forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, ***MongoDB*** guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (***MongoDB*** utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. ***MongoDB*** soporta la búsqueda por campos, consultas

de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del documento, pero también puede ser una función JavaScript definida por el usuario. Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales. Tecnológicamente, ***MongoDB*** es una base de datos multiplataforma y orientada a documentos que brinda alta rendimiento, alta disponibilidad y escalabilidad fácil.

MongoDB trabaja en concepto de colección y documento. La Tabla 7.2 muestra la relación de la terminología de bases de datos relacional respecto a MongoDB.

Tabla 7.2: . Diferencias entre terminología en bases de datos relacionales y MongoDB.

Bases de datos relacionales	MongoDB
Bases de datos	Bases de datos
Tabla	Colección
Tupla o Fila	Documento
Tabla Join	Documentos embebidos
Primary Key	Por defecto *key_id*, gestionada por MongoDB

Asimismo, ***MongoDB*** proporciona una función ***MapReduce*** que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación. El marco de trabajo de agregación permite realizar operaciones similares a las que se obtienen con el comando SQL *"GROUP BY"*. El marco de trabajo de agregación está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los cuales estos datos son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado (véase ejemplo esquemático en Figura 7.3. Todo este procesado es capaz de utilizar índices si existieran y se produce en memoria.

7.6.2. Plataforma tecnológica para el caso práctico

Para la realización del caso práctico se utilizará ***Atlas***, un servicio en la nube gratuito para manegar bases de datos *MongoDB*. ***MongoDB Atlas*** es fácil de configurar y tiene conjuntos de datos de muestra para ejemplos de **R** con *MongoDB*. Puede cargar conjuntos de datos de muestra usando el "..." junto al botón de colecciones en la página de su clúster. No obstante, aunque se puede crear un clúster específico en Atlas, en este ejemplo práctico se parte de uno ya creado.

- Adicionalmente, a modo de apoyo se recomienda utilizar un cliente *MongoDB* para conectarse a la base de datos e inspeccionar los datos contenidos. Esto será extemadamente útil para realizar las consultas. Puede considerarse Robo 3T. Además, si se crea el propio cluster en *Atlas*, este tiene una interfaz amigable para inspeccionar los datos.
- Además de estas funciones, existe documentación de las colecciones y la información contenida en esta base de datos de ejemplo.

Figura 7.3: Ejemplo esquemático del pipeline de agregación en MongoDB. Adaptado de [@Morgan2015]

- Para la resolución de ejercicios puede consultar el Manual de MongoDB que contiene ejemplos y explicación de la sintáxis de *MongoDB*

7.6.2.1. Paquetes R utilizados

El controlador R MongoDB preferido, `mongolite`, es rápido y tiene una sintaxis similar a la del shell MongoDB. `Mongolite` es la que se utilizará en los siguientes ejemplos. Los otros paquetes enumerados aquí no han estado tan activos en Github recientemente. Los paquetes más populares para conectar *MongoDB* y **R** son:

- `mongolite`: un controlador R MongoDB más reciente, *mongolite* puede realizar varias operaciones como indexación, canalizaciones de agregación, cifrado TLS y autenticación SASL, entre otras. Está basado en el paquete jsonlite para R y mongo-c-driver. Podemos instalar mongolite desde CRAN o desde RStudio. `RMongo`: RMongo fue el primer controlador R MongoDB con una sencilla interfaz **R** MongoDB. Tiene una sintaxis como la del shell MongoDB. RMongo ha quedado obsoleto a partir de ahora.

- `rmongodb`: rmongodb tiene funciones para crear pipelines, manejar objetos BSON, etc. Su sintaxis es muy compleja en comparación con mongolite. Al igual que `RMongo`, rmongodb ha quedado obsoleto y no está disponible ni se mantiene en CRAN.

Para poder usar el paquete mongolite debemos cargarlo previamente con el comando siguiente, además de importar la librería previamente.

```
library(mongolite)
```

7.6.3. Conexión y acceso a MongoDB desde R

La variable *cadena_conexion* representa la cadena de conexión a **MongoDB** en **Atlas**. Se podría sustituir por otro servidor o clúster en Atlas si se desea, o un servidor local.

```
cadena_conexion <-
↪  "mongodb+srv://user01:user01@cluster0.mcblc3z.mongodb.net/test"
```

Se establecen en opciones seguridad la no validación de certificados SSL, para evitar que que exista un error de conexión a Atlas.

```
opciones_conexion <- ssl_options(weak_cert_validation = T)
```

NOTA
En entornos reales de producción, está desaconsejado enviar esta comprobación por razones de seguridad.

Después de establecer la conexión a MongoDB se recupera la colección *trips* usando la función ***mongo()*** en código R para obtener la colección de viajes de la base de datos *sample_training*. Esta recopilación contiene datos de viajes realizados por los usuarios de un servicio de bicicletas compartidas con sede en la ciudad de *Nueva York*.

```
viajes <- mongo(collection = "trips", db = "sample_training", url =
↪  cadena_conexion, options = opciones_conexion)
```

Se puede verificar que el código ahora esté conectado a la colección verificando el número total de documentos en esta base de datos. Para hacerlo, use la función `count()`.

```
viajes$count()
#> [1] 10000
```

7.6.4. Obtención de datos en R desde MongoDB

Ahora que hay una conexión establecida con la base de datos, se pueden leer los datos de la misma para ser procesados por **R**. Para mostrar cómo recuperar datos de *MongoDB* y mostrar los mismos se continua con la colección anterior *trips_collection*, mostrada en la sección anterior. Se puede usar la interfaz de usuario de ***MongoDB Atlas*** para ver los documentos

de *trip_collection* o directamnte para visualizarlos. Para ello se puede otener cualquier documento de muestra de la colección usando la función [$iterate().$one()] `iterate()` y `one()` para examinar la estructura de los datos de esta colección.

Una vez se conoce la estructura de los documentos, se pueden realizar consultas más avanzadas, como buscar los 3 viajes más largos a partir de los datos de recopilación de viajes, y luego enumerar la duración en orden descendente. La consulta propuesta utiliza operadores de clasificación y límite para producir este conjunto de resultados.

```
viajes$find(sort = '{"tripduration" : -1}', limit = 3)
#>   tripduration start station id          start station name end station id
#> 1       326222              391         Clark St & Henry St            310
#> 2       279620             3165 Central Park West & W 72 St           3019
#> 3       173357             3155     Lexington Ave & E 63 St           3083
#>           end station name bikeid   usertype birth year
#> 1      State St & Smith St  18591 Subscriber       1979
#> 2        NYCBS Depot - DEL  17547   Customer
#> 3 Bushwick Ave & Powers St  15881   Customer
#>   start station location.type start station location.coordinates
#> 1                      Point                -73.99345, 40.69760
#> 2                      Point                -73.97621, 40.77579
#> 3                      Point                -73.96649, 40.76440
#>   end station location.type end station location.coordinates
#> 1                     Point              -73.98913, 40.68927
#> 2                     Point              -73.98193, 40.71663
#> 3                     Point              -73.94100, 40.71248
#>            start time           stop time
#> 1 2016-01-01 01:58:20 2016-01-04 20:35:23
#> 2 2016-01-02 17:07:26 2016-01-05 22:47:46
#> 3 2016-01-02 15:25:36 2016-01-04 15:34:53
```

7.6.5. Analizando datos de MongoDB en R

Para analizar datos de *MongoDB* con **R** con más detalle, puede usar el marco de agregación de MongoDB. Este marco permite a los operadores crear canalizaciones de agregación que ayudan a obtener los datos exactos con una sola consulta.

Imagínese que se desea verificar cuántos suscriptores realizaron viajes de una duración > 240 segundos y regresaron a la misma estación donde comenzaron. La consulta usa MongoDB $expr para comparar dos campos en el mismo documento.

```
query <-
↪  viajes$find('{"usertype":"Subscriber","tripduration":{"$gt":240},"$expr":
↪  {"$eq": ["$start station name","$end station name"]}}')
```

Combinando estos operadores con código *R*, también se puede ver qué tipo de usuarios son más comunes: *suscriptores* o *clientes únicos*. Para ello, se puede agrupar usuarios por el campo tipo de usuario (*usertype*).

```
tipos_usuario <- viajes$aggregate('[{"$group":{"_id":"$usertype", "Count":
↪  {"$sum":1}}}]')
```

Para comparar los resultados, se pueden visualizar los datos (ver 7.4. Es conveniente convertir los datos obtenidos de `mongolite` en un marco de datos y, por ejemplo, usar el paquete `ggplot2`, para trazar estos datos.

```
library(ggplot2)

df <- as.data.frame(tipos_usuario)

ggplot(df, aes(x = reorder(`_id`, Count), y = Count)) +
  geom_bar(stat = "identity", color = "blue", fill = "green") +
  geom_text(aes(label = Count)) +
  coord_flip() +
  xlab("Tipo de usuario")
```

Figura 7.4: Suscripción por tipo de usuario.

RESUMEN

En este capítulo se ha presentado el concepto de Big Data, porque surje y que aporta respecto a soluciones previas. En particular se ha discutido que son las bases de datos NoSQL y cuales son sus diferencias con las bases de datos relacionales (más tradicionales). Posteriormente se ha explicado la importancia de los métodos de integración de datos, y como ejemplo se mostró la integración de datos en **R** desde *MongoDB*, explicando cómo poder acceder a este tipo de datos y como analizarlos para el caso concreto de una base de datos documental como *MongoDB*.

Capítulo 8

Gobierno y gestión de calidad de datos

Ismael Caballero, Ricardo Pérez del Castillo y Fernando Gualo

8.1. Introducción

Los datos se han convertido en un elemento vital para el desarrollo económico de las organizaciones, ya que permiten una mayor eficiencia en el uso de los recursos y un aumento de su productividad. Tanto es así, que la Unión Europea establece a través de la Estrategia Europea de Datos que en 2030, se establecerá un Espacio Único Europeo de Datos para fomentar un ecosistema con nuevos productos y servicios basados en los datos. Para ello, en esta Estrategia Europea de Datos -que prevé un incremento del 530 % del volumen global de datos- se reclama la necesidad de implantar **mecanismos de gobierno de datos** a través de políticas y directrices consensuadas a alto nivel para alcanzar los objetivos de la **estrategia organizacional** y satisfacer aspectos regulatorios tanto genéricos (como las leyes europeas GPDR o Data Governance Act, o las españolas ENS o ENI) como aspectos sectoriales específicos (como Solvencia II para el sector seguros, o Basilea para el sector financiero).

Estos mecanismos de gobierno de datos deben abordar aspectos verticales relacionados con la adquisición, tenencia, compartición, uso y explotación de los datos en los procesos de negocio, abordando a la vez aspectos transversales relacionados con su gestión: calidad de los datos, aspectos éticos y privacidad, interoperabilidad, gestión del conocimiento y el control sobre los activos de datos a través de las políticas correspondientes, y despliegue de estructuras organizativas con una conveniente separación de los roles de gobierno de datos de los de gestión de datos (ISO, 2017). Por tanto puede decirse que el **gobierno de datos** marca la dirección de cómo la organiación debe realizar la **gestión de datos** para alcanzar los objetivos establecidos en la(s) estrategia(s) de datos de la organización. Esto se consigue mediante la definición de una serie de políticas de datos.

8.2. Concepto de Gobierno de datos

El gobierno de los datos se ha convertido en un habilitador de la economía de los datos (Engels, 2019; Weber et al., 2009), así como también en un pilar básico para la mejora de la transparencia y eficiencia de las administraciones públicas (OECD, 2019; Osimo et al., 2020; Osorio-Sanabria et al., 2020). Aunque existen algunas aproximaciones académicas y profesionales al gobierno de datos, no hay una definición consensuada de este concepto que permita aunar las distintas visiones. No obstante, la definición más aceptada de gobierno de datos es la propuesta por DAMA en DMBoK2 (DAMA, 2017): "*Es la colección de prácticas y procesos que ayudan a asegurar la gestión formal de los activos de datos dentro de una organización mediante el ejercicio de autoridad, control y toma de decisiones compartidas (planificadas, monitorizadas y forzadas*". Teniendo en cuenta los matices que introduce, es también interesante la lectura de la propuesta por Soares (2015), que define **Gobierno de datos** como *"la formulación de políticas para optimizar, securizar, y potenciar los datos como activos organizacionales mediante la alineación de los objetivos de diferentes funciones organizacionales; por su naturaleza, el gobierno de datos requiere cooperación interdepartamental para entregar oportuna y fielmente datos con el máximo valor para la toma de decisiones en la organización"*.

De alguna manera se podría entender que gobernar los datos implica **el diseño, implementación y mantenimiento de un sistema de gobierno de datos**. Es posible afirmar que el gobierno de datos tiene tres características destacables (Caballero et al., 2022b):

- **Está dirigido por el valor de los datos**, pues el principal objetivo del gobierno de los datos es asegurar que los datos son tratados como activos de datos y que la gestión y uso que se hace de ellos permite alcanzar el valor organizacional que se espera de ellos. Por tanto, todas las acciones están encaminadas a la obtención de este valor organizacional.
- **Está centrado en la arquitectura empresarial**, para poder gobernar los datos adecuadamente, es preciso revisar o incorporar ciertos componentes a la arquitectura empresarial tal que o bien den el soporte adecuado o bien forme parte del resultado del gobernar los datos.
- **Es iterativo e incremental**, pues para alcanzar un estado en el que se considere que los activos de datos están perfectamente gobernados es preciso trazar una hoja de ruta que contemple a través de la ejecución de diferentes proyectos relacionados entre sí el desarrollo y la puesta en valor de los artefactos típicos de un sistema de gobierno de datos (véase sección 8.2.2). Esto sólo se puede conseguir en incrementos relevantes (p.ej. la creación de más componentes del sistema de gobierno de datos o la inclusión de nuevos datos a ser gobernados en el alcance de gobierno de datos) para que la organización avance hacia la optimización del valor de los datos.

Un aspecto interesante es que a medida que se avanza en estas hojas de ruta, más sensible se vuelve la organización hacia la importancia de los datos, más aprende la organización a gestionar y gobernar los datos, y más amplio es el alcance del gobierno de datos: en definitiva se puede decir que más madura se vuelve la organización en lo que se refiere al gobierno y a la gestión de los datos.

8.2.1. Beneficios del Gobierno de Datos

Cuando se desarrolla un sistema de gobierno de datos, se espera conseguir con cada incremento uno o una combinación de los siguientes tipos de beneficios (ISACA, 2019):

(1) **Alineamiento estratégico**: Optimización del valor organizacional de los datos mediante el alineamiento con la estrategia organizacional;
(2) **Realización de beneficios**: Aseguramiento de que los datos son entregados en condiciones aceptables a los diferentes consumidores de datos.
(3) **Optimización de Riesgos**: Evitación o minimización dentro del apetito de riesgo de la organiación de los riesgos relacionados con la adquisición, uso y explotación de los datos, asegurando el cumplimiento de la normativa interna y regulatoria; y
(4) **Optimización de Recursos**: Optimización de las capacidades de los recursos humanos y tecnológicos necesarios y utilizados para dar un soporte más eficiente a las distintas operaciones que involucran a los datos, minimizando el desperdicio de recursos al gestionar, usar y explotar los datos.

Estos beneficios deben especificarse como parte de la estrategia de datos de la organización. Así por ejemplo, una organización que considere realizar un alineamiento estratégico y una optimización de riesgos, estará desarrollando una **estrategia defensiva** que debería implementarse a través de un **gobierno técnico**; por otro lado, si una organización quiere por ejemplo maximizar la realización de beneficios, podría considerarse que estaría trazando una **estrategia ofensiva** que se podría materializar mediante un **gobierno para el valor**.

8.2.2. Artefactos de un sistema de Gobierno de Datos

Para poder obtener los beneficios descritos anteriormente, las organizaciones deben realizar esfuerzos para implantar los mecanismos de gobierno de datos reclamados en la Estrategia Europea de Datos particularizándolos a su realidad y en función de su madurez. Estos mecanismos implican el desarrollo de un **sistema de gobierno de datos**, que involucra la creación o mantenimiento de forma interrelacionada y sujeto a las restricciones correspondientes de una serie de artefactos. Dependiendo de si se tiene un gobierno técnico o un gobierno para el valor, la creación y uso de los artefactos será más o menos intensiva. Estos artefactos son los siguientes (Caballero et al., 2022b):

1. **Procesos de Gestión de Datos, Gestión de Calidad de Datos y Gobierno de Datos**, que se refieren al diseño y posterior particularización e implantación de las buenas prácticas relacionadas con las tareas típicas de los datos a nivel de las correspondientes disciplinas. Es posible obtener descripciones genéricas de estos proceso en diferentes modelos de referencias de procesos, tales como **Data Maturity Model (DMM)**(Mecca et al., 2014), **The Data Management Capability Assessment Model (DCAM)**(Council, 2020) o el **Modelo Alarcos de Mejora de Datos (MAMD)**(Caballero et al., 2020)
2. **Estructuras organizacionales**, que deben recoger las cadenas de responsabilidades y rendición de cuentas, haciendo una adecuada separación entre las responsabilidades propias del gobierno de datos así como las responsabilidades propias de gestión de datos y

las de gestión de calidad de datos. Los roles que deben asumir estas responsabilidades son típicamente el de *Chief Data Officer* (Soares, 2015; Treder, 2020) con un punto de vista más ejecutivo/estratégico y los de *data stewards* (Plotkin, 2020) desde una perspectiva más táctica/operativa.

3. **Principios, políticas y marcos de referencia**, que deberían incluir todos los principios rectores en los que se basará el uso de los datos (tales como los *Generaly Accepted Information Principles* listados en Ladley (2019)), las directrices o políticas y los controles correspondientes asociados necesarios para modelar y gestionar el valor de los datos, el riesgo a asumir y las restricciones a considerar según se describe en ISO/IEC 38505-2 (ISO, 2018).
4. **Datos**, que se refiere tanto a los datos que se deben gobernar como a las descripciones necesarias a través de los metadatos correspondientes. Para la parte de datos es fundamental poder establecer una adecuada **arquitectura de datos** con los correspondientes modelos que recojan la semántica del entorno de la organización y refleje el conocimiento de cómo ésta usa los datos para desarrollar su actividad organizacional y/o económica. Para dar soporte al uso correspondiente deben generarse y mantenerse los metadatos correspondientes, que pueden ser de varios tipos (DAMA, 2017): (1) *metadatos de negocio* recogidos típicamente en glosario de negocio y que describen la relación del dato con el negocio; (2) *metadatos técnicos* recogidos habitualmente en los *catálogos de datos* que describen detalles técnicos de los datos; y (3) *metadatos operacionales* recogidos típicamente en los diccionarios de datos que recogen aspectos relacionados con el procesamiento y acceso a los datos. Es importante que todos estos metadatos estén reconciliados convenientemente entre ellos ya que su visión conjunta permitirá conocer una descripción adecuada de los datos que se pretenden gobernar, y si está descripción es suficiente como para poder usar los datos con la suficiente garantías de éxitos.
5. **Cultura, ética y comportamiento**, cuyo objetivo es identificar aquellos aspectos culturales y éticos que deben regular la forma en la que la organización abordará las tareas relacionadas con los datos para que estos tengan el valor organizacional deseado (Harrison et al., 2019).
6. **Personas, habilidades y competencias**, que trata de organizar los roles que deben asumir las diferentes responsabilidades relacionadas con los diferentes procesos; también debe enfocarse en asegurar que esos roles tienen los conocimientos, habilidades y competencias necesarias para abordar las tareas asociadas mediante los programas formativos correspondientes; finalmente, este artefacto incluye asegurar que la organización tiene planes de contingencia ante la eventual rotación funcional de los recursos humanos dedicados a las responsabilidades relacionadas con los datos (Plotkin, 2020).
7. **Servicios, infraestructuras y aplicaciones** que aborda todo lo relacionado con las tecnologías y sistemas de información para dar soporte a las diferentes actividades de los procesos de gestión de datos, gestión de calidad de datos y gobierno de datos

8.3. Marcos y metodologías existentes de Gobierno de Datos

En la literatura, tanto académica como profesional, existen algunas propuestas de creación de sistemas de gobierno de datos. Es interesante resaltar que en el ámbito académico se han desarrollado algunas revisiones sistemáticas de literatura científica para identificar los componentes del gobierno de datos, aunque de forma general, y salvo algunas referencias, se mantienen desconectados de las propuestas profesionales. También es importante mencionar que salvo COBIT 2019, la inmensa mayoría de estos marcos no identifican explícitamente el concepto de "**sistema de gobierno de datos**", sino que se establece bajo un paraguas más genérico de "**gobierno de datos**". En cualquier caso, la idea es la misma. En los siguientes párrafos se resumen los aspectos más importantes de los marcos más relevantes, que pueden servir para que el lector encuentre el que mejor se adapta a su circunstancia personal:

- Abraham et al. (2019) analizan la literatura para identificar los elementos de un marco de trabajo teórico que clasifican en torno a cuatro áreas del gobierno de datos: (1) alcance organizacional (aspectos intra e inter organizacionales), (2) alcance de los datos (datos tradicionales vs big data), (3) alcance del dominio (calidad de datos, seguridad de datos, arquitectura de datos, ciclo de vida, metadatos, almacenamiento e infraestructura de datos), y (4) Mecanismos de gobierno (estructurales, procedimentales y relacionales).
- Al-Ruithe et al. (2019) identifican a través de una revisión sistemática de literatura identifican las áreas o retos del gobierno de datos donde merece la pena llevarse a cabo investigación. Éstas son tecnología (seguridad, privacidad, disponibilidad, rendimiento, clasificación de los datos, migración de datos), legalidad, y aspectos organizacionales o del negocio.
- Brous et al. (2016) derivan, basándose en una revisión sistemática de la literatura, los principios para desarrollar de forma efectiva estrategias y aproximaciones para el gobierno de datos, que agrupan en torno a cuatro conceptos fundamentales: (1) organización, (2) alineamiento, (3) cumplimiento y (4) entendimiento común de los datos.
- Carruthers and Jackson (2020) identifican los posibles elementos que deben contemplarse en la transformación digital, la cual debe apoyarse en el gobierno de datos. Estos elementos (personas, datos, procesos, tecnologías) son representados mediante un triángulo, en el que los datos están en el centro de este. En la continuación de la obra de estos autores presentada en Jackson and Carruthers (2019), se propone un modelo de transformación, convenientemente soportado en el gobierno de datos.
- DCAM (Council, 2020) es un modelo de referencia para la evaluación de la capacidad de gestión de datos desarrollado por EDM Council. El modelo tiene ocho componentes agrupados en cuatro niveles (1) Fundamentos (Estrategia de Datos y Casos de negocio; Programas de Gestión de Datos y financiación), (2) Ejecución (Arquitectura de datos y de negocio; Arquitectura de datos y de tecnología; Gestión de Calidad de datos; Gobierno de Datos), (3) Colaboración (Entorno de Control de Datos) y (4) Formalización del diseño e implementación de las actividades analíticas.
- DMBoK (DAMA, 2017) es un marco de referencia de procesos desarrollado por DAMA que posiciona el gobierno de datos como la función que guía el resto de las acciones relacionadas con la gestión de datos. Identifica una serie de elementos que deben generarse a partir

del gobierno de datos: estrategia de gobierno de datos; estrategia de datos; hoja de ruta del gobierno de datos; principios de gobierno de datos, políticas de gobierno de datos, procesos; marco operativo de gobierno de datos; hoja de ruta y guía de implementación; plan de operaciones; glosario de términos; plan de operaciones; cuadro de mandos de gobierno de datos; etc.

- El Data Management Maturity Model (DMM) (Mecca et al., 2014) es un modelo desarrollado por el SEI. Introduce un modelo de referencia para la evaluación de la madurez de las prácticas de gestión de los activos organizacionales de gestión de datos y sus correspondientes actividades a través del ciclo de vida de estos. Este modelo se puede usar de forma no prescriptiva para guiar a las organizaciones en la implantación de las mejores prácticas de gobierno y gestión de datos.
- Eryurek et al. (2021) identifica los "ingredientes" propios de un sistema de gobierno de datos (Herramientas; personas y procesos; cultura de datos) así como las áreas en las que debería enfocarse el gobierno de datos a lo largo del ciclo de vida de los datos (descubrimiento y curación de datos; gestión de datos; políticas de privacidad, seguridad y acceso
- COBIT 2019 (ISACA, 2019) identifica para el sistema de gobierno de tecnologías y de información los siguientes elementos: procesos; estructuras organizacionales; principios, políticas y marcos de referencia; información; cultura, ética y comportamiento; personas, habilidades y competencias; servicios, infraestructuras y aplicaciones.
- ISO 38505-1 (ISO, 2017) e ISO 38505-2 (ISO, 2018) muestran los aspectos claves del gobierno de datos (valor de los datos, riesgo, y restricciones) e introducen seis principios (responsabilidad, estrategia, adquisición, rendimiento, cumplimiento y comportamiento humano) junto con unas guías de desarrollo teniendo en cuenta una serie de factores que determinan su implementación. Identifica una serie de procesos (evaluar, dirigir, monitorizar) como áreas propias de actuación del gobierno de datos, distinguiéndolos y estableciendo las relaciones correspondientes con las operaciones propias de la gestión de datos. Sin embargo, no describe actividades propiamente dicho para la creación de sistemas de gobierno de datos.
- Khatri and Brown (2010) aduce que el gobierno de datos implica tomar decisiones sobre activos claves de datos en varios dominios de decisión (principios de datos, calidad de datos, metadatos, acceso a datos y ciclo de vida de los datos).
- Janssen et al. (2020) explora las capacidades de gobierno de datos que las organizaciones dirigidas por datos necesitan para extraer el máximo beneficio de los sistemas algorítmicos basados en Big Data (*Big Data Algorithmic Systems*) y propone un marco para el gobierno de datos para optimizar estos sistemas.
- Ladley (2019) presenta un marco de gobierno de datos basado en cinco pilares: compromiso, estrategia, arquitectura y diseño, implementación y operación, y por último, cambio.
- Lillie and Eybers (2019) estudian la literatura existente para identificar los aspectos más interesantes sobre (1) el alcance y los constructos más importantes del gobierno y gestión de datos, (2) las capacidades ágiles que son requeridas en el gobierno y la gestión de datos y (3) la necesidad de las organizaciones africanas, incluyendo las administraciones públicas, para establecer capacidades ágiles en las funciones de gobierno y gestión de datos.
- En el *Proceso Unificado de Gobierno de Datos* de IBM (Soares, 2010) se identifican cinco "*ingredientes claves*" en el gobierno de los datos que deberían ser cubiertos por cualquier

marco: (1) Fuerte respaldo por parte de la organización con soporte de las TI, (2) Centrarse en los elementos de datos críticos, (3) Énfasis en los artefactos de datos, (4) Alineación en torno a métricas y aplicación de políticas, y (5) celebración de las victorias rápidas conseguidas como hitos en una hoja de ruta a largo plazo.

- La OECD recoge en su informe sobre gobierno de datos para administraciones públicas (OECD, 2019) las mejores prácticas que diferentes administraciones de distintos países han seguido para lograr la transparencia de las prácticas de gobierno y para incrementar el valor de los datos de la ciudadanía de cara a una mejor prestación de servicios públicos.
- Treder (2020) identifica algunos componentes específicos que debería tener un sistema de gobierno de datos (cadenas de valor; estrategia de dato; procesos de datos; descripción de los roles y sus responsabilidades; gestión del equipo de la oficina del dato), así como las áreas en las que debe enfocarse el gobierno de datos (casos de negocio; aspectos éticos y cumplimiento; gestión y análisis de datos).

8.3.1. Modelo Alarcos de Mejora de Datos (MAMD)

A modo de ejemplo, se van a introducir más detalles sobre un marco de referencia basado en estándares internacionales ISO: el **Modelo Alarcos de Mejora de Datos (MAMD) v3.0** (Caballero et al., 2020). MAMDv3.0 es un marco de trabajo que se usa para la evaluación y mejora de la capacidad de los procesos de la organización relacionados con la gestión, la gestión de la calidad de los datos y el gobierno de datos. Tiene dos componentes principales:

- El **modelo de referencia de procesos (MRP)**, que contiene una descripción de los procesos de gestión de datos, de gestión de calidad de datos y de gobierno de datos. Este MRP está alineado con los principales estándares en el área (ISO 8000-61(ISO, 2016), e ISO/IEC 38505-2 (ISO, 2018)), así como con las buenas prácticas de otros modelos como DAMA, DMM o COBIT 2019. la Fig.8.1 representa el modelo de referencia de procesos

- El **modelo de evaluación de procesos (MEP)**, sigue las directrices de evaluación y niveles de capacidad y madurez descritas por ISO/IEC 33000 y adaptadas a la evaluación de procesos de datos conforme al propuesto en ISO 8000-62 (ISO, 2018) la Fig. 8.2 representa el modelo de madurez incluidos.

8.4. Gestión de calidad de datos

Datos con niveles inadecuados de calidad, acaban teniendo un impacto negativo para las organizaciones, bien en términos económicos, bien en términos de reputación (Redman, 2016).Por eso es importante que las organizaciones cuiden del nivel de calidad de sus datos, y se aseguren que dichos niveles permanecen dentro de los niveles permitidos para que la ejecución de los procesos de negocio se haga dentro del apetito de riesgo de la organización.

Se dice que un conjunto de datos tiene calidad cuando sirven para el propósito para el que fueron recogidos (*fitness for use*) (Strong et al., 1997b). Para determinar si un conjunto de

Figura 8.1: Modelo de Referencia de Procesos de MAMD

Figura 8.2: Modelo de Madurez de MAMD

datos tiene calidad suficiente para dicho propósito, es preciso identificar y seleccionar un conjunto de criterios (llamados en la literatura **dimensiones** (Wang, 1998), o **características de calidad de datos** (ISO/IEC, 2008a)) que van a permitir representar los requisitos de calidad de datos que tienen los usuarios relevantes para el contexto de uso de dichos datos. Al conjunto de dimensiones o características de calidad de datos seleccionadas se les llama **modelo de calidad de datos**. Por tanto, puede decirse, que para determinar la calidad de los datos debe elegirse un modelo de calidad de datos. En la tabla mostrada en la Fig. 8.3 se introduce una descripción de las características de calidad de datos incluidas en el estándar ISO/IEC 25012. Como puede observarse estas características se clasifican en dos grandes bloques: **Inherentes** y **Dependientes del Sistema**. Las **inherentes** se refieren al grado con el que las características de calidad de los datos tienen un potencial intrínseco para satisfacer las necesidades establecidas y necesarias cuando los datos son utilizados bajo condiciones específicas; las **dependientes del Sistema**, por otro lado, permiten determinar el grado con el que la calidad de datos es alcanzada y preservada a través de un sistema informático cuando los datos son utilizados bajo condiciones específicas. Se puede ampliar más información sobre este aspectos en Caballero et al. (2017).

Para ilustrar el significado de algunas de estas características (p.ej. **exactitud**, **completitud**, o **consistencia**), a continuación se introducen algunos ejemplos:

- Como ejemplo de niveles inadecuados de **exactitud sintáctica** -en este caso- podría ponerse el hecho de un atributo `Nombre` de la entidad `Persona` toma un valor "*Marja*" (no existente en los datos de referencia de nombre) en lugar de "*María*" (que sí que está incluido).
- El hecho de que un atributo `Nombre` de la entidad `Persona` toma el valor de "*George*" en vez de "*Jorge*" para almacenar datos de la Persona llamada realmente "*Jorge*" es un ejemplo de nivel inadecuado de **exactitud semántica**. Ambos valores son sintácticamente correctos, pero `George` es otra persona distinta a `Jorge`, y quien guardó los datos, simplemente se equivocó de persona.
- Supóngase que para una determinada aplicación, se necesitan recoger valores para los siguientes atributos de una entidad Persona `DNI`, `Nombre`, `Apellido1`, y `Apellido2` para ser usada adecuadamente en un contexto de uso. En caso de faltar alguno de ellos (falta de completitud), podría ocurrir que los datos de la persona no fueran usables; inculso, podrían faltar algunos atributos más como por ejemplo `email`, pero si no son relevantes para la aplicación, daría igual no tenerlo.
- Un ejemplo de falta de consistencia podría darse cuando el valor del atributo `FechaNacimiento` de la entidad Persona no es anterior o igual a la fecha de hoy.

[**GEMA!!**: como no he conseguido disminuir el tamaño de la letra en la tabla con RMarkDown, he puesto una imagen con la tabla, creo que ahora queda más o menos bien. Una de las "consecuencias" es que en el caption aparece la palabra "figura" y no "tabla"]

Diferentes autores han proporcionado diferentes mecanismos para medir y evaluar la calidad de los datos usando las dimensiones o características de calidad seleccionadas. Aunque para dar soporte a este proceso, se han propuesto numerosas metodologías de evaluación (Batini et al., 2016), el principal problema de estas contribuciones es que, típicamente, se han realizado *ad hoc* y no permiten ni generalizar los resultados obtenidos ni compararlos con los obtenidos por

Característica	Definición	Inherente	Dependiente del sistema
Exactitud	Grado en que los datos tienen atributos que representan correctamente el valor de un atributo	X	
Completitud	Grado en el que existen suficientes valores para todos los atributos necesarios para la representación de una entidad	X	
Consistencia	Grado en que los datos están libres de contradicción y son coherentes con el resto de los datos de su contexto de uso	X	
Credibilidad	Grado en que los datos se consideran ciertos y creíbles por los usuarios	X	
Actualidad	Grado en que los datos tienen atributos con las fechas y tiempos correctos	X	
Accesibilidad	Grado en que los datos pueden ser accedidos por cualquier usuario	X	X
Cumplimiento	Grado en el que los datos están construidos conforme estándares, convenciones o regulaciones	X	X
Confidencialidad	Grado en el que los datos tienen atributos específicos que solo pueden ser accedidos por usuarios autorizados	X	X
Eficiencia	Grado en el que los datos tienen atributos que pueden ser procesados y provistos dentro de los niveles de rendimiento esperados	X	X
Exactitud	Grado en el que los datos tienen atributos que son precisos	X	X
Trazabilidad	Grado en el que los datos tienen atributos que proveen información detallada sobre los cambios realizados en los datos	X	X
Comprensibilidad	Grado en el que los datos se expresan de manera que los usuarios puedan leerlos e interpretarlos correctamente	X	X
Disponibilidad	Grado en el que los datos están disponibles para ser accedidos por usuarios y/o aplicaciones autorizadas		X
Portabilidad	Grado en el que los datos pueden ser alojados, reemplazados o movidos desde un sistema a otro		X
Recuperabilidad	Grado en el que los datos disponen de formas de mantener un nivel especificado de operabilidad incluso cuando se producen fallos		X

Figura 8.3: Tipos de datos

otras organizaciones (Loshin, 2011). Para paliar estos problemas, se han desarrollado estándares que recogen los conocimientos y principios básicos comunes para medir y evaluar la calidad de los datos. Ejemplos de estos estándares pueden ser la mencionada ISO/IEC 25012 (ISO/IEC, 2008a), o como ISO 8000-8 (ISO/IEC, 2015) que recogen características de calidad, ISO/IEC 25024 (ISO/IEC, 2008b) que recoge aspectos específicos de cómo llevar a cabo las mediciones de las características, o ISO/IEC 25040 (ISO/IEC, 2011) que proporciona una metodología de evaluación de calidad del software que puede ser adaptada a la evaluación rigurosa y sistemática de la calidad de los datos.

Es preciso en este punto introducir la principal diferencia entre **medir** y **evaluar** la calidad: medir consiste en determinar la cantidad de calidad de datos que tiene un conjunto de datos; mientras que evaluar implica determinar si, de acuerdo al apetito de riesgo de la organización, la cantidad de calidad de datos medida es suficiente y adecuada para usar los datos en el contexto de uso establecido por los datos. En cualquier caso, la evaluación de calidad de datos requiere primero medir la calidad. Y para medir la calidad, primero deben definirse procedimientos de medición que ayuden a conseguir información no sólo de la cantidad de calidad de datos que tiene un dataset, sino también de cómo esta cantidad de calidad de datos medida representa aspectos concretos de la calidad. En ese sentido, ISO 25024 (ISO/IEC, 2008b) proporciona una serie de propiedades medibles para cada una de las características presentadas en la Tabla representada en la Fig. 8.3; además para cada una de estas propiedades medibles, el estándar proporciona un método de medición genérico, que permitirá, convenientemente particularizado medir dichas propiedades y luego agruparlas para determinar el valor de la característica de calidad de datos. En la Fig.8.4 se muestran las propiedades medibles para las características de

calidad identificadas como inherentes.

Figura 8.4: Propiedades de las características inherentes de calidad de datos

Una de las ventajas de usar estas propiedades medibles es que es posible identificar mejor qué está causando que los datos no tengan los niveles de calidad adecuados y por tanto actuar directamente sobre ellas.

Así por ejemplo para medir el grado de exactitud de un conjunto de datos, habría que medir las propiedades "*Exactitud Sintáctica*", "*Exactitud Semántica*" y "*Rango de Exactitud*", y luego hacer algún tipo de agrupación de los resultados que tuviera en cuenta la importancia o peso relativo de cada una de estas propiedades a la hora de evaluar la Exactitud. Supóngase, por ejemplo que fuera significativo usar una media ponderada para realizar la agregación de las mediciones de las propiedades medibles, representando α, β, γ coeficientes representando el peso relativo de cada una de las propiedades. Con esto se podría usar la fórmula mostrada en la ecuación (8.1) para el cálculo de la **exactitud**.

$$Exactitud = \alpha \times ExactitudSemántica + \beta \times ExactitudSintática + \gamma \times RangoExactitud \quad (8.1)$$

A la hora de medir las propiedades medibles correspondientes a las características de calidad de datos, es interesante tener en cuenta que ISO 25024 proporciona procedimientos de medición que, típicamente, están basados en la comprobación de si se violan las reglas de negocio que regulan la adecuación al uso de los datos en un contexto determinado (Loshin, 2002). Así, ISO 25024 proporciona métodos de medición de tipo ratio, que suelen corresponderse con la fórmula mostrada en la ecuación (8.2)):

$$Medida\ PropiedadDQ = \frac{Número\ Registro\ Violando\ Reglas\ de\ Negocio\ Específicas}{Número\ Total\ de\ Registros} \quad (8.2)$$

A la hora de implementar esta fórmula como método de medición, uno de los ejercicios más difíciles es recolectar y validar las reglas de negocio específicas que rigen la validez de los datos.

Siguiendo con uno de los ejemplos anteriores, se va a medir la Exactitud Sintáctica usando la fórmula que propone ISO 25024 y que se representa en la ecuación (8.3)):

$$ExactitudSintática = \frac{Número\ Registros\ que\ Son\ Semánticamente\ Correctos}{Número\ Registros\ Totales} \tag{8.3}$$

Si se tuviera un ejemplo en el que se pretende medir el nivel de exactitud sintáctica del atributo `DNI` se podría usar la siguiente regla de negocio "*el DNI tiene que seguir la especificación para DNI o para el NIE, correspondiente con la expresión regular* `(d{8})([A-Z])`". Para realizar la validación de la reglas de negocio habría que escribir algún tipo de programa que permitiera validar cuántos registros cumplen la regla de negocio establecido; tras ralizar la comprobación de la regla, hay que contar el número de registros que no lo han cumplido y el número de registros totales para poder generar el valor correspondientes de la medida de la propiedad. Como se verá en la siguiente subsección, en algunos entornos académicos y profesionales, alternativamente a la escritura de este programa, se utilizan técnicas de perfilado de datos para realizar mediciones. En muchos casos, esto se consigue asumiendo de forma implícita un enunciado y validación de las reglas de negocio que rigen cuando los datos son adecuados para un contexto de uso.

Para ilustrar cómo se realiza la medición de una característica de calidad, supóngase que una organización, en base a su apetito de riesgo para un determinado proceso de negocio, considera que para una determinada aplicación, puede asignar los siguientes pesos a los coeficientes $\alpha = 0{,}4$, $\beta = 0{,}4$, y $\gamma = 0{,}2$ (de la fórmula de la ecuación (8.1)) y que además, ha obtenido como resultados de la medición de las propiedades medibles de la Exactitud, los siguientes valores: para la `ExactitudSemántica`, 80; para la `ExactitudSintática`, 60; y para el `RangoExactitud`, 70. Entonces, el grado de exactitud del dataset sería 70.

Una vez que se han realizado estas mediciones, el siguiente paso es realizar la evaluación propiamente dicha. Esto consisten en comparar los resultados obtenidos en las mediciones con unos valores umbrales mínimos de aceptación deseables especificados por los usuarios de los datos a través de los requisitos de calidad de datos; es importante resaltar que estos valores umbrales dependen fuertemente del apetito de riesgo de la organización a la hora de utilizar estos datos (Redman, 2016). Siguiendo el ejemplo anterior de la exactitud, suponga el lector que una organización considera que si no se alcanza un valor de al menos 75 % para el uso pretendido de los datos, entonces, éstos no deberían ser usados. En el ejemplo, y como se ha obtenido un resultado de 70 % para la medición, entonces, como se requiere un valor de al menos 75 %, se puede concluir que **los datos no tienen el nivel adecuado de calidad para el uso pretendido, y por tanto no deben usarse**. Esto no significa que los datos no puedan usarse en otro contexto en el que por ejemplo el valor umbral se estableciese en 65 % en función del apetito de riesgo de la organización.

En algunos contextos de uso, como se verá posteriormente en el capítulo **??**, antes de poder usar los datos se realiza un proceso de preparación de datos que tiene como objetivo determinar y adecuar los niveles de calidad al uso que se pretende mediante un proceso de evaluación y mejora de los datos - típicamente llamado **limpieza de los datos**- . En estos contextos, la aproximación presentada basada en características de calidad de datos y propiedades medibles no se suele usar, recurriéndose a una evaluación y limpieza más estadística. Se pierde entonces

de alguna manera la capacidad de establecer una dirección más efectiva y sobre todo alineada a las necesidades reales de la organización de las operaciones de evaluación y limpieza de datos.

Finalmente, es interesante mencionar que, basándose en el estándar ISO/IEC 25012 (ISO/IEC, 2008a) e ISO/IEC 25024 (ISO/IEC, 2008b) es posible certificar el nivel de calidad de datos de un repositorio de datos. (Gualo et al., 2021) recoge experiencias de medición, evaluación y certificación de calidad de datos.

8.4.1. Medición de calidad de datos vs perfilado de datos

En esta subsección se plantea el **perfilado de datos** como una técnica para realizar la medición de las propiedades medibles de las características de calidad de datos. Abedjan et al. (2015) clasifica los tipos de **data profiling** en las siguientes categorías:

- **Perfilado de columna simple**, que implicaría tareas de identificación de cardinalidades, identificación de patrones y tipos de datos, distribución de valores de datos, clasificación de dominios.
- **Perfilado de columnas múltiples**, que implicaría tareas de correlación y reglas de asociación, identificación de clusters y outliers, elaboración de resúmenes de datos y bocetos.
- **Perfilado de dependencias**, que a su vez implica:
  - **Detección de reglas de unicidad** tales como la identificación de claves, identificación de condiciones e identificación de sinónimos.
  - **Detección de dependencias de inclusión**, que puede abarcar el descubrimiento de claves ajenas o la identificación de dependencias condicionales de inclusión.
  - **Dependencias funcionales**, como pueden ser las dependencias condicionales.

En **R Software** se puede utilizar el paquete `dlookr` creado por Choonghyun Ryu y descrito en https://github.com/choonghyunryu/dlookr o aquí y que contiene algunas funciones interesantes que pueden ayudar a realizar determinadas tareas de perfilado. Por ejemplo, la función `overview()` da información general sobre un conjunto de datos; resulta muy interesante la función `diagnose()` que proporciona información realizando un perfilado de los valores únicos y los valores únicos de un conjunto de valores. En el siguiente fragmento se demuestra el tipo de información proporcionada por `diagnose(idealista18::Madrid_POIS$City_Center)`:

```
library(dlookr)
diagnose(idealista18::Madrid_POIS$City_Center)
#> # A tibble: 2 x 6
#>   variables types   missing_count missing_percent unique_count unique_rate
#>   <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
#> 1 Lon       numeric             0               0            1           1
#> 2 Lat       numeric             0               0            1           1
```

Si estas funciones de perfilado proporcionan información suficiente y adecuada, se puede para computar las mediciones de las propiedades medibles. Por ejemplo, se puede utilizar el resultado de la columna `missing count` para calcular el grado de completitud de las variables `longitud`y `latitud`, que se pueden establecer en 100 % al ser `missing count` para las dos variables. Incluso se pueden utilizar funciones como `plot_na_pareto()` para visualizar un gráfico de Pareto mostrando las variables que no tienen valores nulos.

Finalmente es interesante mencionar que el paquete `dlook` incluye funciones como `diagnose_paged_report()` que permiten elaborar informes que contienen información sobre las estructuras de datos del paquete, avisos, descripción de las variables, valores perdidos, valores únicos de las variables categóricas y numéricas, distribuciones de valores ceros y negativos, posibles outliers, ... El siguiente chunk explica cómo crear un informe de 15 páginas en formato PDF con toda esa información sobre la variable `idealista18::Madrid_POIS$Metro` (**OJO LO DEJO COMENTADO PORQUE ME DABA FALLO AL COMPILAR**:

```
# diagnose_paged_report(idealista18::Madrid_POIS$Metro)
```

En ocasiones, y retomando la idea de las reglas de negocio, se puede utilizar este tipo de informes para derivar algunas reglas de negocio sobre los datos que se pueden usar en la medición de las propiedades medibles seleccionadas de las características de calidad de datos que mejor representan los requisitos de calidad de los datos de los usuarios. Teniendo en cuenta que los valores almacenados en una base de datos provienen del esfuerzo realizado durante un tiempo indeterminado, las reglas de negocio encontradas deberían responder a los esfuerzos continuados a lo largo del tiempo por la organización para mantenerse actualizado. Pero en ocasiones ha podido ocurrir que estos mantenimientos no se hayan realizado de la forma más rigurosa ni trazable posible, con lo que sería altamente recomendable validar con el personal del negocio si las reglas halladas o inferidas a partir de estos informes realmente se corresponden con lo que el negocio necesita. A modo resumen puede decirse que la información proporcionada por el perfilado puede usarse en primera instancia para derivar reglas de negocio que no se conocen a partir del estado actual de los datos; y, en segunda instancia para recoger información que se puede emplear durante el proceso de medición de determinadas características de calidad de datos. En el capítulo **??** se profundizará en el proceso de estudio de dos caracteríticas de calidad de datos: completitud y consistencia.

8.4.2. Mejora de datos

Si los datos no tienen el nivel de calidad que se necesita de ellos para una tarea, es preciso mejorar su calidad para que no se arruinen los proceso de negocio. Para ello, a partir de los resultados de las mediciones, los analistas de calidad de datos deben determinar las cuasas raíces de esos niveles inadecuados de calidad de datos. Strong et al. (1997a) identifican diez posibles obstáculos que pueden hacer que los datos podrían no tengan esos niveles adecuados de calidad:

1. Múltiples fuentes de datos producen diferentes valores para el mismo atributo de la misma entidad.

2. La realización de juicios subjetivos en la producción de los datos, puede llevar a valores diferentes.
3. Errores Sistemáticos en la producción de Información llevan a la pérdida de información.
4. Grandes volúmenes de información almacenada dificultan su acceso en tiempo razonable.
5. Sistemas heterogéneos distribuidos llevan a definiciones inconsistentes, formatos y valores.
6. La información no numérica es difícil de indexar.
7. Análisis automatizado de los contenidos en colecciones de información pueden no producir resultados adecuados. 8.A medida que las necesidades de los usuarios para la realización de tareas en entornos organizacionales cambian, la información que es relevante y útil cambia
8. Un acceso fácil a la información puede entrar en conflicto con los requisitos de seguridad, confidencialidad y privacidad.
9. La falta de recursos de computación limita el acceso a los datos en circunstancias favorables.

En función de la naturaleza del problema detectado, las acciones correctivas pueden ser de distinta naturaleza:

- **Corrección de causas sistemáticas**. Si se observa que los problemas se suceden de forma sistemática y repetida, entonces las acciones de mejora de datos deben estar orientada a eliminar esas causas sistemáticas, como las descritas anteriormente en Strong et al. (1997a). Por ejemplo: si los errores de calidad de datos se deben a que un proceso de negocio que está mal diseñado, entonces hay que rediseñarlo; si las causas se deben a que hay personas desempeñando ciertos roles que no tienen los conocimientos o habilidades adecuadas, entonces, hay que darle la formación adecuada; o si se deben a que hay software (por ejemplo, procesos ETL) que fallan, entonces hay que realizar el mantenimiento correctivo correspondientes.

- **Corrección de errores debidos a causas aleatorias**. Si no es posible identificar cuáles son las causas raíces, porque son completamente desconocidas o aleatorias, entonces, no queda más remedio que actuar sobre los valores de los datos, cambiándolos para asegurarse que se cumplen las reglas de negocio que están establecidas. A este proceso se le suele llamar *data cleansing*. Ilyas and Chu (2019) identifican diversas técnicas de limpieza de datos (que pueden incluir operaciones de **imputaciones de datos** - véase la sección 9.3.2 del capítulo **??**, de normalización, o de estandarización): esto implica realizar limpieza basadas en reglas de negocio, deduplicación de datos, transformación de datos, o limpieza guiadas por machine learning. En este caso, sería posible utilizar algunas funciones del paquete `dlookr` relacionadas con la transformación de los datos tales como `imputate_na()` que genera valores estadísticamente significativos para el caso de los valores faltantes o `imputate_outlier()` que genera valores estadísticamente significativos para el caso de outliers (valores con niveles adecuados de precisión o de consistencia).

Es importante resaltar que en caso de que los errores se deban a causas sistemáticas, suele ser preciso incluir también acciones de limpieza de datos para poder devolver el estado de la base de datos a un estado coherente.

Tanto para el caso de la evaluación, como para la limpieza, siempre resulta conveniente (si no necesario en la mayoría de los casos), tener recopiladas y validadas un conjunto de reglas de negocio que se puedan usar en la evaluación y medición de la calidad de los datos (Ilyas and Chu, 2019). En este sentido, en (Caballero et al., 2022a) se proporciona una metodología para la identificación de las reglas de negocio y de cómo deben clasificarse para medir y evaluar determinadas dimensiones de calidad de datos.

RESUMEN

En este capítulo se ha presentado los fundamentos del gobierno de datos. Es importante tener en cuenta los siguientes aspectos:

- El gobierno de datos tiene como objetivo asegurar que los datos que se usan y gestionan en las organiaciones están alineadas a las estrategias de datos de la organización, maximizando así el valor organizacional de los datos.
- Gobernar los datos implica el diseño, implementación y mantenimiento de un sistema de gobierno de datos
- Un sistema de gobierno de datos tiene siete tipos de componentes: procesos de gestión de datos, gestión de calidad de datos y gobierno de datos; estructuras organizacionales; principios, políticas y marcos de referencia; datos y descripción de los datos; cultura, ética y comportamiento; personas, habilidades y competencias; servicios, infraestructuras y aplicaciones.
- Existen modelos de referencias que pueden ser usados como base para la creación de sistemas de gobierno de datos.
- El gobierno de datos persigue cuatro beneficios básicos para la organización: alineamiento estratégico, realización de beneficios, optimización de riesgos, optimización de recursos.
- La gestión de la calidad de datos es el proceso mediante el cual se garantiza que los datos tengan el nivel de calidad adecuado para las tareas para las que fueron pensados.
- Para evaluar y medir la calidad se necesitan criterios que ayuden a medir la calidad de los datos; estos criterios se llaman características o dimensiones de calidad de datos.
- La evaluación y medición de calidad de datos requiere la identificación y clasificación de las reglas de negocio que rigen la validez de los datos
- Las técnicas y herramientas de perfilado de datos se pueden utilizar como base para la realización de las mediciones de calidad de datos; y en ocasiones para la identificación de reglas de negocio a partir de los datos.
- Cuando los datos no tienen calidad, a partir de las mediciones y evaluaciones realizadas, deben investigarse cuáles son las posibles causas. Si las causas son sistemáticas, entonces hay que enfocar el problema desde un punto de vista organizacional; si las causas son aleatorias, se pueden usar las técnicas de limpieza de datos vistas.

Capítulo 9

Integración y limpieza de datos

Jorge Velasco López

9.1. Introducción

En proyectos de modelado predictivo, como la clasificación o la regresión, generalmente es necesario realizar un preprocesamiento previo de los datos. Esto se debe a razones, (Brownlee, 2020), tales como que (i) algunos algoritmos de aprendizaje automático requieren determinados requisitos de datos (por ejemplo, que sean numéricos); (ii) es necesario corregir el ruido estadístico y los errores en los datos; (iii) no pueden extraerse de los datos directamente las relaciones no lineales complejas.

En consecuencia, para asegurar niveles adecuados de calidad y permitir descubrir la estructura subyacente del caso de uso a los algoritmos de aprendizaje, se deben procesar los datos en bruto con anterioridad a poder usarse para probar y evaluar un modelo de *machine learning*. Este paso, en un proyecto de modelado predictivo, se conoce como *preparación de datos*, o bien *preprocesamiento de datos*.

Cada proceso de preparación es muy específico para cada conjunto de datos, para los objetivos del proyecto y para los algoritmos que se utilizarán para modelar los datos. Sin embargo, hay una serie tareas comunes que suelen requerirse en esta fase de preparación de datos, como la **integración de los datos** (combinación de datos de distintas fuentes) y la **limpieza de los datos** (identificar y corregir posibles errores en los datos como las causas de niveles inadecuados de calidad).

Otras tareas que típicamente se incluyen en el proceso de preparación de datos son: 1) la selección de variables (*feature selection*); 2) transformaciones de datos para cambiar la escala o distribución de variables; 3) transformación de variables (*feature engineering*); 4) reducción de dimensionalidad (creación de proyecciones compactas de los datos) y 5) otro tipo de tratamientos.

Este capítulo se centra en la parte de limpieza de datos y en la integración de datos. Los datos recogidos en su origen se denominan **datos sin procesar** o **brutos** y se recopilan en el contexto de un problema que desea resolver. Si se requiere recuperar los datos de diversas fuentes para obtener vistas unificadas se habla de integración de datos. Aunque los datos están típicamente representados en formato matricial, las variables pueden ser de distinto tipo de datos (numéricas, rangos, porcentajes, categóricas, binarias, etc.). Además, debe tenerse en cuenta que muchos algoritmos de aprendizaje automático fundamentalmente solo operan con datos numéricos.

La colección de paquetes `Tidyverse` (Wickham and Wickham, 2017) dispone de una serie de herramientas destinadas a facilitar la manipulación, importación, exploración y visualización de datos y se utiliza exhaustivamente en ciencia de datos. Está compuesto de los siguientes paquetes: `readr, dplyr, ggplot2, tibble, tidyr, purr, stringr` y `forcats`, que ya se han introducido en buena medida en anteriores capítulos. La mayoría de las funciones del paquete `tidyverse` crean objetos *tibble*, que son compatibles con los `data.frame` por ser objetos muy similares (tabulares, organizados en filas y columnas) aunque hay alguna pequeña diferencia.

El paquete `caret` (Kuhn, 2008), por su parte, proporciona una interfaz unificada que simplifica el proceso de modelado. Para ello, emplea la mayoría de los métodos de aprendizaje estadístico implementados en **R Software**, además de numerosas funciones para realizar varias tareas del proceso de modelado: *feature selection*, *data splitting*, validación del modelo, etc. En relación al preprocesamiento de los datos, incluye la función `preProcess(x, method)`, que se puede integrar en el entrenamiento (función `train()`) para estimar los parámetros de las transformaciones a partir de la muestra de entrenamiento y posteriormente aplicarlas automáticamente al hacer nuevas predicciones. El parámetro `method` permite establecer una lista de procesamientos, para imputación, creación y transformación de variables explicativas, selección de predictores y extracción de componentes. El paquete `caret` también incluye una serie de funciones auxiliares, como `rfe()` y `dummyVars()`.

Las principales utilidades de `tidyverse` en relación con la manipulación de ficheros y variables se han ilustrado en el capítulo 4. En este capítulo, se revisarán los problemas de calidad de las variables, se observará cómo manejar valores atípicos en variables cuantitativas y el tratamiento de valores *missing*, para lo que se usará el paquete de `caret`. Además, se dará una introducción al manejo del ruido estadístico con el paquete de **R** `stat`. Como fuente de datos, se utilizará el set de datos de `Gapminder`, que contiene información demográfica de todos los países desde 1952, como la cantidad de población (variable `pop`), expectativa de vida (`lifeExp`) e ingresos per cápita (`gdpPercap`), asi como el país (`country`), continente (`continent`) y año de referencia (`year`). Por último, se dará una visión de los procesos de integración de datos y de la correcta asignación del tipo de dato, para lo que se usará el set de datos de la empresa de comercio electrónico "Beauty eSheep". Tanto el set de datos de `Gapminder`como éste, están incluidos en el paquete `CDR`.

9.2. Problemas de calidad de datos

Es habitual que en el tratamiento de algunas de las variables numéricas, se presenten problemas de calidad de datos (véase capítulo 8). Para ilustrarlo en el set de datos `Gapminder`, podríamos

sospechar que la variable numérica `gdpPercapita`, estuviera en una medida (moneda) distinta para determinados continentes en determinadas condiciones. Por ejemplo, para los países que tienen un valor de `lifeExp` menor de 60. Para verificar esa hipótesis, se podría hacer uso de funciones ad-hoc o manipulaciones en los datos que permitan obtener un *subset* de los datos de interés bajo determinadas condiciones, o bien crear variables derivadas que faciliten esta tarea. En este ejemplo, se podría agrupar por continente y año y se podría presentar en orden descendiente un resumen de los principales descriptivos [1].

```
gapminder_cons <- gapminder |>
  mutate(pib_filtro = ifelse(lifeExp < 60, gdpPercap * pop / 10^9, NA)) |>
  group_by(continent, year) |>
  summarize(
    med_gdpPercap = mean(gdpPercap),
    dt_gdpPercap = sd(gdpPercap),
    media_pob = mean(pop),
    media_pib_filtro = mean(pib_filtro),
    dt_pib_filtro = sd(pib_filtro)
  ) |>
  arrange(desc(media_pib_filtro))
# Para obtener un tibble desagrupado habría que añadir la función ungroup()
head(gapminder_cons, n = 2)
#> # A tibble: 2 x 7
#> # Groups:   continent [1]
#>   continent  year med_gdpPercap dt_gdpPercap media_pob media_pib_filtro
↪  dt_pib~1
#>   <fct>     <int>         <dbl>        <dbl>     <dbl>            <dbl>
↪  <dbl>
#> 1 Africa     1957         1385.        1135.  5093033.             7.36
↪  14.5
#> 2 Africa     1952         1253.         983.  4570010.             5.99
↪  11.4
#> # ... with abbreviated variable name 1: dt_pib_filtro
```

Para el caso de variables cualitativas, siguiendo el ejemplo con el mismo set de datos, podría ocurrir que hubiera algún valor de la variable `continent` con espacios en blanco y con problemas de mayúsculas y minúsculas. A continuación, se presenta una posible actuación para este ejemplo.

```
continent <- as.character(gapminder$continent) # Casteo de la variable a tipo
↪  carácter
continent <- str_trim(continent) # Eliminación de espacios en blanco
```

[1] Como se puede comprobar, una simple combinación de `mutate()` y la condición lógica `ifelse` facilita el filtrado solo allí donde se necesita, en este caso en el momento de crear la variable derivada de `pib_filtro`. Si se confirmara la hipótesis, deberían realizarse las oportunas correcciones para las filas afectadas. En nuestro ejemplo se observa que para el continente africano en la década de los 50 presentan valores bajos de PIB promedio, consistente con la evidencia y por tanto no serían necesarias correcciones posteriores.

```
gapminder$continent <- as.factor(str_to_upper(continent)) # Convertir a
↪  mayúsculas
levels(gapminder$continent) # Previsualizacion de datos corregidos
```

La función `mutate()` también puede servir para cambiar el tipo de dato, como en el siguiente ejemplo, donde se utiliza conjuntamente con la función `across()` para abarcar todas las columnas tipo `factor` del `data.frame`.

```
# Convertir todos los factores en tipo carácter
Gapminder_conv <- gapminder |> mutate(across(where(is.factor), as.character))
```

Además del uso de las funciones de transformación de ficheros y variables como las presentadas en `tidyverse`, la *visualización* también puede ayudar en la identificación de problemas de calidad. En la Fig. 9.1, se usa la función `facet_wrap()` para hacer diagramas de caja agrupados con las variables `x=continent` y `y=gdpPercapita`. A continuación, se muestran los puntos con `geom_jitter()` y se agrega `facet_wrap` como capa con el `continent` como argumento.

```
gapminder |>
  filter(continent %in% c("Americas", "Europe")) |>
  filter(year %in% c(1952, 1982, 2002)) |>
  ggplot(aes(x = factor(year), y = gdpPercap, fill = continent)) +
  geom_boxplot() +
  geom_jitter(width = 0.1, alpha = 0.2) +
  xlab("Año") +
  facet_wrap(~continent, ncol = 2) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Figura 9.1: Visualización boxplot

Por su parte, un ***outlier*** es un registro que no cumple una serie de reglas de negocio especificadas para ese atributo, es decir, aquel que tiene niveles inadecaudos de exactitud o de consistencia (véase capitulo @(DGDQM)). Una buena forma de identificar los valores atípicos (*outliers*) en variables cuantitativas, es a través de un **boxplot** como el indicado en la Fig. 9.1, ya que en éste se puede identificar como están representados los datos con respecto a sus métricas descriptivas (media, mediana, cuartiles). De alguna forma, se está realizando un perfilado de datos para descubrir reglas de negocio, que son descritas en este caso, bajo un fundamento estadístico. No obstante, como se comentaba en el capitulo @(DGDQM), dichas reglas deberían ser validadas con la gente de negocio para poder tener la certeza de que realmente responden a lo que necesita el negocio. Por ejemplo, se podría considerar que podrían ser *outliers* algunas observaciones de baja `pop` y alto `gdpPercapita`. El análisis exploratorio mostraría que son valores atípicos, pero que no son incorrectos (es decir, tienen niveles adecuados de calidad):

```
atipicos <- gapminder |> filter(gapminder$gdpPercap > 40000, gapminder$pop <
↪  1500000)
head(atipicos, n = 2)
#> # A tibble: 2 x 6
#>   country continent  year lifeExp    pop gdpPercap
#>   <fct>   <fct>     <int>   <dbl>  <int>     <dbl>
#> 1 Kuwait  Asia       1952    55.6 160000   108382.
#> 2 Kuwait  Asia       1957    58.0 212846   113523.
```

9.3. Niveles inadecuados de completitud: Valores *missing*

9.3.1. Visualización

Los datos pueden faltar por multitud de posibles razones. Sin embargo, estas razones generalmente se agrupan en dos categorías: **valores *missing* informativos** (Kuhn et al., 2013) y **valores *missing* aleatorios** (Little and Rubin, 2019). Los informativos implican una causa estructural, ya sea por deficiencias en la forma en que se recopilaron los datos o por anomalías en el entorno de observación. Los aleatorios implican que los valores faltantes ocurren independientemente del proceso de recopilación de datos. La categoría que causa los valores nulos determinará cómo se deben manejar. Por ejemplo, se puede dar a los informativos su propia categoría (por ejemplo, "Ninguno"), ya que su valor único puede afectar el rendimiento predictivo. Mientras que los aleatorios pueden implicar la eliminación o la imputación. Además, los diferentes algoritmos de aprendizaje automático manejan la falta de información de manera diferente. De hecho, la mayoría de los algoritmos no incorporan mecanismos para manejarlos (por ejemplo, modelos lineales generalizados y derivados, redes neuronales y SVM) y, por lo tanto, requieren que se traten previamente. Solo unos pocos modelos (principalmente basados en árboles) tienen procedimientos incorporados para tratar los valores nulos.

En **R**, los valores nulos se representan con el símbolo NA (*Not Available*, "no disponible") y los valores indefinidos (p. ej., dividir por cero) se representan con el símbolo NaN (*Not a Number*,"no es un número"). La función `vis_miss()` en el paquete de **R** `visdat` (Tierney,

2017) también permite una fácil visualización de patrones de datos nulos, con opciones de clasificación y agrupación. Los valores nulos se indican mediante una celda negra. Las variables y los patrones NA se han agrupado por filas (es decir, clúster = TRUE) en la Fig. 9.2.

```
# gapminder_cons se le ha imputado algunos valores missing
vis_miss((gapminder_cons), cluster = TRUE)
```

También el paquete `dlookr` permite realizar informes de distribución de valores nulos.

Figura 9.2: Visualización de valores missing

9.3.2. Técnicas de Imputación

La gestión de los valores *missing* debe hacerse considerando la problemática que se quiera resolver. Una primera opción a considerar sería excluir los valores **missing**, si bien se estaría eliminando información. Si se quisiseran filtrar los registros nulos para alguna columna se podría utilizar la función `is.na()`.

```
gapminder |>
  filter(!is.na(pop))
# En el ejemplo, se filtran los casos que no son NA de la variable `pop`.
# También se puede usar la función `drop_na()` del paquete `tidyr`
```

También se puede optar por cambiar el valor del NA, por ejemplo por un 0, de la siguiente manera:

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

No obstante, estas dos últimas, no son acciones recomendadas en primera instancia, porque filtrar registros con valores nulos o introducir valores que podrían no respetar la semántica de los datos puede ocasionar un alto impacto negativo en los niveles globales de calidad de datos del dataset.

Se puede ir más allá de la eliminación de valores *missing* o de la estandarización. A través de diversos métodos se pueden generar valores que, con mayor o menor probabilidad, podrían ser los que realmente correspondieran a estos valores faltantes. A estas técnicas se las conocen como **técnicas de imputación de valores**. Para imputar valores perdidos se puede usar la función `preProcess()`del paquete `caret` o la función `imputate_na()` del paquete `dlookR`. En el primer caso, hay diferentes valores que se le puede pasar al parámetro *method*:

- `knnImpute`: permite utilizar el algoritmo kNN (k-vecinos más cercanos) para imputar valores perdidos. El algoritmo kNN requiere que se le indique el número de vecinos a utilizar en la predicción a través del parámetro k;
- `bagImpute`: con este valor se usan varios árboles de decisión para hacer la imputación del valor *missing*;
- `medianImpute`: imputa la mediana para el caso de una variable numérica, que suele ser preferible a imputar la media, puesto que el promedio puede verse afectado por *outliers*. A continuación, se indica cómo funciona cada método de imputación usable la función `preProcess()`del paquete `caret`:

```
# Se convierte a data.frame
gapminder_cons <- data.frame(gapminder_cons, stringsAsFactors = FALSE)
# Se realiza el preprocesamiento:
pre_knn <- preProcess(gapminder_cons, method = "knnImpute", k = 2)
pre_bag <- preProcess(gapminder_cons, method = "bagImpute")
pre_median <- preProcess(gapminder_cons, method = "medianImpute")
# Se obtienen los datos
imputed_knn <- predict(pre_knn, gapminder_cons)
imputed_bag <- predict(
  pre_bag,
  gapminder_cons
)
imputed_median <- predict(pre_median, gapminder_cons)
```

9.4. Mejorando la exactitud y la precisión: eliminación del ruido estadístico

El **ruido estadístico** es una variabilidad no explicada dentro de una muestra de datos. Generalmente consiste en errores (errores de medición y errores de muestreo) y residuos (diferencia entre su valor observado y el valor predicho).

R tiene, entre otras, dos funciones útiles para filtrar y reducir el ruido, **filter()** y **fft()** del paquete `stats`. Para explorar el uso del primero, en la Fig. 9.3 se visualizan los dos conjuntos de datos que se usarán como ejemplos: una señal con ruido y una señal pura superpuesta sobre un fondo exponencial.

Figura 9.3: Señal con ruido

La función `stat::filter()`aplica el filtrado lineal a una serie temporal univariante o a cada serie por separado de una serie temporal multivariante. Para estimar la relación señal-ruido, se puede utilizar el máximo de la señal pura y la desviación estándar de la señal con ruido usando 100 puntos divididos uniformemente entre los dos extremos.

La función `filter()` de **R** toma la forma general `filter(x, filter)`, donde `x` es el objeto que se filtra y `filter` es un objeto que contiene los coeficientes del filtro.

Para crear un filtro de media móvil de siete puntos idénticos, cada uno igual a 1/7, se puede usar la función `rep()`. Con ello se obtienen los valores del artículo original (**?**) y luego se aplica a la señal con ruido. Con este filtro, la relación señal-ruido mejora de 4,57 a 14,28 como se puede comprobar en la Fig. 9.4.

```
medmov_7 <- rep(1 / 7, 7)
# La aplicación de este filtro a nuestra señal ruidosa devuelve el siguiente
↪  resultado
señal_ruido_medmov_7 <- stats::filter(señal_ruido, medmov_7)
puntuacion2 <- max(señal_gauss) / sd(señal_ruido_medmov_7[c(1:50, 201:250)],
↪  na.rm = TRUE)
```

9.5. Integración de datos

La integración de datos es una combinación de procesos técnicos y de negocio que se utilizan para combinar información proveniente de diferentes fuentes para convertirla en datos fiables y

Figura 9.4: Señal con tratamiento del ruido

valiosos. En términos generales, se puede decir que consiste en acceder a los datos desde todas las fuentes y localizaciones tanto en entorno local, en la nube o de una combinación de ambos y la integración de datos, de modo que los registros de una fuente de datos mapeen registros en otra. Con ello, se obtenienen las vistas necesarias para el proceso de análisis o de negocio.

En el capítulo 4 se introducía el concepto de las uniones de transformación y de filtrado. En este se desarrollará mediante un ejemplo práctico, junto a otros procesos como la comprobación de la integridad referencial y de las claves principales (o primary keys -PK-) y otras transformaciones, sobre el set de datos que se usarán en los capítulos correspondientes al *machine learning*, correspondientes a la empresa de comercio electrónico "Beauty eSheep". Las tablas principales de su *datawarehouse* son:

- `dim_pedido`: Pedidos realizados por los clientes. Con las siguientes columnas: ide_pedido (factor), des_pedido (factor).
- `dim_fecha_venta`: Distintas fechas en las que se han producido ventas en la empresa. Columnas: ide_fecha_venta (factor), des_fecha_venta (factor).
- `dim_vendedor`: Proveedores. Columnas: ide_vendedor (factor), des_vendedor (factor).
- `dim_categoria_productos`: Catálogo de categorías de productos de la empresa. Columnas: ide_categoria_productos (factor), des_categoria_productos (factor).
- `dim_producto`: Detalle de los distintos productos que conforman el catálogo de ventas de la empresa. Columnas: ide_producto (factor), des_producto (factor), ide_categoria_productos (factor).
- `dim_nivel_educacion_cliente`: Niveles de educacación de los clientes. Columnas: ide_nivel_educacion_cliente (tipo factor), des_nivel_educacion_cliente (factor).
- `dim_cliente`: Clientes que tiene o ha tenido la empresa. Columnas: ide_cliente (factor), ide_cliente_nif (factor), des_cliente (factor), num_edad (entero), num_tamano_familiar (entero), num_anos_experiencia (entero), num_ingresos_ano (doble), ide_nivel_educacion_cliente (factor).

Una de las herramientas principales para la integración, son las uniones de transformación, que

agregan las columnas de un conjunto y a otro x enlazando las filas, según las claves:

- `inner_join()`: incluye todas las filas en x e y.
- `left_join()`: incluye todas las filas en x.
- `right_join()`: incluye todas las filas en y;
- `full_join()`: incluye todas las filas en x o y. Si una fila en x enlaza a varias filas de y, se devuelven todas las filas de y, por cada vez que enlace con x.

Por otro lado, las uniones de filtrado filtran las filas de x en función de la presencia o ausencia de coincidencias en y:

- `semi_join()` devuelve todas las filas de x con una coincidencia en y;
- `anti_join()` devuelve todas las filas de x que no tengan una coincidencia en y. La función
- `nest_join()` devuelve todas las filas y columnas de x con una nueva columna anidada, que contiene todas las coincidencias de y. Cuando no hay ninguna coincidencia, la columna de la lista es un tibble de 0 filas.

Figura 9.5: Proceso de integración de datos iniciales

En la Fig. 9.5, se muestra el proceso de integración de los datos iniciales de "eSheep". En el siguiente código se puede ver parte de dicho proceso, donde se aplican los conceptos vistos hasta ahora:

```
# Se hace para todas las dimensiones.
# Por ejemplo, para la tabla fac_venta: se convierten en factores las
↪  variables
fac_venta <- fac_venta |>
  mutate_at(vars((starts_with("ide"))), as.factor)
```

```
# Asimismo, se reconvierte la variable ide_fecha a tipo fecha:
fac_venta$ide_fecha <- as.Date(fac_venta$ide_fecha, "%d/%m/%Y")
if (any(is.na(fac_venta$ide_fecha))) {
  warning(paste("Fechas ide_fecha erroneas en fac_venta"))
}
# Comprobación de la clave primaria o primary key (pk) para la tabla
↪  fac_venta:
if (nlevels(fac_venta$ide_venta) != nrow(fac_venta)) {
  stop(paste(
    "Error PK fac_venta filas: ", nrow(fac_venta),
    " valores distintos", nlevels(fac_venta$ide_venta)
  ))
}
# Comprobación de la integridad referencial entre fac_venta y dim_fecha_venta
df_error <- fac_venta |>
  anti_join(dim_fecha_venta)
if (nrow(df_error)) {
  stop(paste("No integridad ref fac_venta vs DIM_FECHA_VENTA, n. filas
  ↪  erróneas: ", nrow(df_error)))
}
# En este paso se ha usado la función anti_join, que sirve para identificar
↪  los registros de una tabla que no coinciden con la otra tabla.
# A continuación, se construye el tablón con los datos agregados de las
↪  compras de los clientes
dim_cliente$ide_cliente <- as.factor(dim_cliente$ide_cliente)
dp_fac_cliente_venta <- fac_venta |>
  group_by(ide_cliente, ide_producto) |>
  summarise(num_imp = sum(num_imp), ind_pro = as.logical(sum(num_imp))) |>
  ungroup() |>
  inner_join(dim_producto) |>
  select(ide_cliente, des_producto, num_imp, ind_pro) |>
  pivot_wider(
    names_from = des_producto, names_sort = T,
    values_from = c(ind_pro, num_imp),
    values_fill = list(ind_pro = FALSE, num_imp = 0)
  ) |>
  mutate_if(is.logical, funs(factor(., levels = c("FALSE", "TRUE"), labels =
  ↪  c("N", "S")))) |>
  inner_join(dim_cliente) |>
  inner_join(dim_nivel_educacion_cliente) |>
  select(-c("ide_nivel_educacion_cliente")) |>
  arrange(as.character(ide_cliente))
# En la función de pivot_wider, se puede observar que se usa el parámetro
↪  values_fill, que rellena donde las combinaciones no existen.
# Por su parte,  mutate_if(is.logical, funs(factor(.,
↪  levels=c("FALSE","TRUE"), labels=c("N","S")))) sustituye FALSE por N y
↪  TRUE por S.
```

Como resultado del proceso de integración, se obtiene la tabla dp_fac_cliente_venta (a partir de la que se obtiene dp_fac_cliente_venta_pro13), con los campos que se listan a continuación:

- `ide_cliente (tipo factor)`: Código identificador único del cliente.
- `ind_pro11-17 (factor)`: Indicador si el cliente es consumidor de ese producto ('S') o no ('N').
- `importe_pro11-17 (doble)`: Importe neto global consumido por el cliente en ese producto en euros.
- `ide_cliente_nif (factor)`: Código identificador oficial único del cliente.
- `des_cliente (factor)`: Descripción del cliente (apellidos, nombre).
- `edad (entero)`: Edad del cliente.
- `tamano_fam (entero)`: Número de miembros de la unidad familiar a la que pertenece el cliente.
- `anos_exp (entero)`: Años de trabajo del cliente.
- `ingresos_ano (doble)`: Ingresos anuales del cliente en euros.
- `des_nivel_edu (factor)`: Descripción del nivel de educación del cliente.

En general, los tipos de datos más comunes son los numéricos y los categóricos, aunque cada uno puede subdividirse aún más, como tipo entero, doble y punto flotante para variables numéricas y booleano, ordinal o nominal para variables categóricas. El tipo `factor` de **R**, es de tipo categórica con un número finito de valores o niveles.

A partir de `dp_fac_cliente_venta` y `dp_fac_cliente_venta_pro13`, se obtiene `dp_entr` (datos para entrenamiento de los modelos en su escala original) y `dp_test` (datos para testeo de los modelos en su escala original). A su vez, a partir de estos útimos, se obtienen los equivalentes con variables sólo numéricas (`dp_entr_num` y `dp_test_num`), que son los conjuntos de datos preparados para la fase de modelado. En el próximo capítulo, se puede ver cómo se obtienen y cómo usarlos en el resto del preprocesamiento.

RESUMEN

- En un proyecto de modelado predictivo, como la clasificación o la regresión, los datos en bruto normalmente no pueden usarse directamente, sino que deben procesarse para asegurar niveles adecuados de calidad y permitir descubrir la estructura subyacente del caso de uso a los algoritmos de aprendizaje.
- Se han visto las principales funciones de manipulación de datos del paquete `tidyverse` con el propósito de limpieza de datos y el paquete `caret` para la imputación de valores missing.
- Se ha dado una visión del tratamiento de los datos cuando se presentan problemas de calidad de datos, valores *missing* y problemas de ruido estadístico.
- Se ha realizado un proceso de integración de datos con el ejemplo de "Beauty eSheep".

Capítulo 10

Análisis exploratorio de datos

Emilio L. Cano

10.1. Introducción

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

10.1.1. El cuarterto de Anscombe

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

```
library(dplyr)
anscombe |> summarise(across(.fns = mean))
#>  x1 x2 x3 x4       y1     y2     y3     y4
#>   9  9  9  9   7.5009  7.5009  7.5   7.5009
anscombe |> summarise(across(.fns = sd))
#>    x1    x2       x3       x4     y1      y2    y3      y4
#> 3.316  3.316   3.316   3.316   2.031   2.031   2.030   2.030
```

Sin embargo, la Fig. 10.1 muestra que, a pesar de tener medias y desviaciones típicas prácticamente iguales, los datos son muy diferentes.

Figura 10.1: Representación de las variables del cuarteto de Anscombe

Si en el análisis por separado ya se ve la necesidad de hacer un gráfico, ésta es más evidente cuando se analizan las variables conjuntamente. La Figura 10.2 muestra los cuatro gráficos que constituyen "El cuarteto de Anscombe", y que se puede obtener de la propia ayuda del conjunto de datos (`example(anscombe)`). La línea de regresión que se ajusta es prácticamente la misma, y los coeficientes de correlación entre las variables X e Y de los cuatro gráficos, idénticos: 0.8163. Es evidente que la relación entre las variables es muy distinta en cada uno de los casos, y si no se visualizan los datos para elegir el mejor modelo de regresión y después interpretarlo, se pueden tomar decisiones erróneas. El cuarteto de Anscombe es muy ilustrativo, al igual que *The Datasaurus Dozen:* (Matejka and Fitzmaurice, 2017) en https://www.autodeskresearch.com/publications/samestats.

10.1.2. Conceptos generales

Muy brevemente, se presentan una serie de conceptos esenciales para la mejor comprensión de este manual[1]. Los datos que se analizan, provienen de una determinada **población**, y no son más que una **muestra**, es decir, un subconjunto de toda la población. La **Estadística Descriptiva** se ocupa del AED en sentido amplio, que se aplica sobre los datos concretos de la muestra. La **Inferencia Estadística** (véase Capítulo 13 hace referencia a los métodos mediante los cuales, a través de los datos muestrales, se toman decisiones, se analizan relaciones, o se hacen predicciones sobre la población. Para ello, haremos uso de la **Probabilidad** (véase Capítulo 12) aplicando el método más adecuado. Además, será muy importante considerar el método de obtención de la muestra (véase Capítulo 14 que, en términos generales, debe ser representativa de la población para que las conclusiones sean válidas. La Fig. 10.3 representa la esencia de la Estadística y sus métodos.

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

Figura 10.2: Los cuatro gráficos que constituyen El cuarteto de Anscombe junto con un ajuste lineal

Figura 10.3: La esencia de los métodos estadísticos

Por otra parte, los individuos de una población vendrán clasificados por un carácter o fenómeno estadístico. Éstos pueden ser de dos tipos:

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

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

10.1.3. Componentes de un gráfico y su representación en R

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

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

1. Proporcionar una **tabla de datos** a la función `ggplot`. Es el primer argumento (`data`) y se puede utilizar el operador *pipe*.

2. Proporcionar las **columnas** de la tabla de datos que serán representadas en el gráfico. Este será el segundo argumento (`mapping`) de la función `ggplot`, y se especifica con la función `aes` (*aesthetics*) como una lista de pares *aesthetic* = *variable*, de forma que el elemento especificado como *aesthetic* será "mapeado" a los valores de la variable. Esta especificación se puede hacer también en las funciones que añaden capas, que se explican a continuación. Los *aesthetics* más comunes (y para muchos tipos de gráficos obligatorios) son `x` e `y`, es decir, las variables que se usarán para el eje horizontal y el eje vertical respectivamente. Además, se pueden especificar variables para el color, el tamaño, el símbolo de los puntos, el tipo de línea, el texto, y otros específicos del tipo de gráfico. Los *aesthetics* se pueden especificar también de forma "fija" (sin depender de ninguna variable) fuera de la función `aes`.

3. Añadir las **capas** del gráfico con los *geoms*, es decir, los objetos geométricos que representan a cada variable. Esto se indica con el operador `+`, como si se "sumasen" componentes al gráfico mediante funciones `geom_xxx`.

4. Añadir otras capas al gráfico, por ejemplo, una capa de etiquetas del gráfico (función `labs`), de ejes, para modificar los ejes y leyendas creados por defecto (funciones `scale_*_xxx`), para crear nuevas variables a representar basadas en los datos (funciones `stat_xxx`).

5. Añadir un tema al gráfico, por ejemplo, en blanco y negro, o con especificaciones concretas como el posicionamiento de la leyenda.

6. Añadir "facetas" (*facets*). De esta forma se divide el gráfico en varios subgráficos basándose en los valores de una o más variables discretas (normalmente categóricas).

10.2. Análisis exploratorio de una característica

10.2.1. Variables cualitativas

El resumen numérico de variables cualitativas se muestra en la tabla de frecuencias, la cual se puede representar con un gráfico de barras o con un gráfico de sectores[2]. Por ejemplo, el conjunto de datos `accidentes2020_data` disponible en el paquete `CDR`, describe los datos de accidentes de tráfico en la Ciudad de Madrid registrados por Policía Municipal con víctimas y/o daños al patrimonio. Entre sus variables, contiene la variable cualitativa tipología del accidente `tipo_accidente`. Un resumen puede obtenerse tanto con la función `table()` como con el paquete `dplyr`, como se vio en la Sección 4.6: En variables cualitativas, llamamos a la categoría más frecuente **moda** de la variable.

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

Para representar el gráfico de barras con la función `ggplot()` se añade la función `geom_bar()` (ver Fig. 10.4).

```
library(ggplot2)
accidentes2020_data |>
```

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

```
ggplot() +
geom_bar(aes(y = tipo_accidente), fill = "pink")
```

Figura 10.4: Gráfico de barras con ggplot2

El código anterior es la forma más básica de hacer un gráfico con `ggplot2`. Opciones más avanzadas pueden encontrarse en el libro *R for Data Science* (Wickham and Grolemund, 2016).

Ya se ha comentado que los gráficos de sectores no se recomiendan a menos que se incluya información numérica. El paquete `ggstatsplot` realiza gráficos que incluyen análisis estadísticos. Por ejemplo, la función `ggpiestats()` proporciona un gráfico de sectores con algunos tests estadísticos (ver la ayuda de la función) y podría utilizarse para determinar en qué medida un conjunto de 80 ayuntamientos de distinto signo político presta o no un determinado servicio `serv` (ver el conjunto de datos en el paquete del libro `?CDR::ayuntam`). El siguiente código produce el gráfico de la Figura 10.5.

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

Una alternativa a los gráficos de sectores son los *waffle charts* (gráficos de gofre, o de tableta de chocolate). La siguiente expresión produce el gráfico de la Figura 10.6 usando el paquete `waffle`. Con el argumento `use_glyph` se pueden incluir iconos en vez de cuadrados.

```
library(waffle)
freq <- ayuntam |>
```

Figura 10.5: Gráfico de sectores con tests. Prestación o no de un determinado servicio X

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

Figura 10.6: Gráfico waffle: Prestación o no de un determinado servicio X por parte de 80 ayuntamientos

10.2.2. Variables cuantitativas

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

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

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

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

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

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

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

```
#>              MAD    2742.81
#>              IQR    3776.25
#>               CV       0.29
#>         Skewness       1.82
#>      SE.Skewness       0.01
#>         Kurtosis       5.77
#>          N.Valid   49576.00
#>        Pct.Valid      89.69
```

La representación de la tabla de frecuencias de una variable continua es el **histograma**. En el caso de las variables continuas, para representarlo, se cuenta el número de observaciones (frecuencia) por intervalos (*bins*). Una posible regla sería el método de Sturges[3], que se puede hallar con la función `nclass.Sturges()`.

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

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

Sin embargo, esta regla no siempre es la más apropiada, como se verá en la Sección 40.3, pues debe estudiarse bien la naturaleza de la variable a analizar.

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

La función `geom_histogram()` del paquete `ggplot2` añade una capa con un histograma al gráfico. El color de las barras se controla con el *aesthetics* `fill`, y la altura puede representar las frecuencias absolutas (recuentos) o relativas (proporciones). El número de intervalos se indica con el argumento `bins`, o alternativamente la anchura de intervalo con `bin_width`, véase la Fig. 10.7.

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

h1 <- p + geom_histogram(color = "yellow", fill = "pink")
```

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

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

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

Figura 10.7: Histogramas de la renta neta media per cápita en 2019 con distintos bins. Iquierda: bins por defecto (n=30); Centro: bins con la regla de Strurges; Derecha: bins = 20

Una representación alternativa al histograma es la línea de densidad, que sustituye las barras por una línea continua, generalmente suavizada. A continuación, se añade la linea de densidad a uno de los histogramas de la Fig. 10.7.

```
p + geom_histogram(aes(y = after_stat(density)),
  position = "identity",
  color = "yellow", fill = "pink"
) +
  geom_density(lwd = 1, colour = 4)
```

Otra representación gráfica muy útil de las variables continuas son el gráfico de **cajas y bigotes** y el **diagrama de violín**, que se obtienen fácilmente combinando en `ggplot()` las capas `geom_boxplot()` y `geom_violin()` respectivamente (véase Fig. ??).

Figura 10.8: Histograma y linea de densidad de la renta neta per capita española en 2019

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

Figura 10.9: Boxplot y violin plot de la renta neta media por persona (€) en 2019

Otra visualización básica para una variable numérica es la visualización secuencial de las obser-

vaciones, bien a través de puntos (`geom_point()`) o a través de líneas (`geom_line()`). El orden de las observaciones nos pueden indicar cuándo se ha producido un cambio u otros patrones.

10.3. Análisis exploratorio de varias características

En el apartado 10.2 se ha realizado un AED de variables aisladas, pero lo usual es incluir las relaciones entre otras variables dentro del AED. Además, en el caso de las variables numéricas se pueden obtener algunas medidas de resumen conjunto que indiquen la relación entre las distintas características. Las herramientas estadísticas utilizadas son: (i) las tablas de frecuencias conjuntas, que, en el caso de dos atributos, pueden ser tablas de doble entrada, con un atributo en filas y el otro en columnas, para determinar si existe asociación entre las variables, como se verá en el Capitulo 23; (ii) los resúmenes numéricos, como la covarianza, el coeficiente de correlación, coeficientes de asociacion, etc. y (iii) los gráficos en los que se puede representar más de una variable.

10.3.1. Variables cualitativas

El resumen numérico sigue siendo la tabla de frecuencias, en este caso conjuntas para los distintos niveles que toman las variables. Para dos atributos, se puede representar en forma de tabla de doble entrada. Este tipo de tabas se denominan **tablas de contingencia** (ver Capitulo 23).

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

```
table(ayuntam$signo_gob, ayuntam$serv)
#>
#>              No Sí
#>   Avanzados  14 28
#>   Ilustrados  6 32
```

Aunque la representación gráfica más habitual siguen siendo los gráficos de barras:

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

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

frecuencias + proporciones
```

Figura 10.10: Grafico de barras de la prestación pública del servicio *X* por parte de 80 Ayuntamientos de distinto signo político. Iquierda: frecuencias absolutas. Derecha: frecuencias relativas.

Una visualización interesante de tablas de doble entrada son gráficos en los que se representan las frecuencias conjuntas por medio de puntos cuyo área es proporcional a la frecuencia. La Fig. 10.11 muestra gráficamente la tabla de frecuencias conjunta de los atributos `signo_gob` y `serv` del conjunto de datos `ayuntam`.

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

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

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

Figura 10.11: Representación gráfica de tabla de frecuencias con la función 'ballonplot()'

En cualquier caso, se pueden representar más variables creando "subgráficos" o facetas (*facets*). Basta con añadir una capa al gráfico `ggplot2` con la función `facet_wrap` y argumento `facets` una lista de variables (categóricas o discretas) para cuyos valores queremos hacer un gráfico distinto.

```
accidentes2020_data |>
  ggplot(aes(sexo, fill = estado_meteorológico)) +
  facet_wrap(vars(tipo_accidente)) +
  geom_bar()
```

10.3.2. Variables cuantitativas

La descripción conjunta de variables numéricas se puede resumir con el vector de medias (media de cada variable) y la matriz de varianzas-covarianzas. La covarianza es una medida del grado de dependencia lineal entre dos variables numéricas. Si la covarianza es cero, no hay relación lineal (pero podría ser otro tipo de relación, recuérdese el cuarteto de Anscombe). Pero la covarianza es una medida que depende mucho de la escala de las variables, y es más fácil interpretar el coeficiente de correlación lineal (función `cor()`), que está acotado entre -1 y 1. Cuanto más se acerque a 1 en valor absoluto, más fuerte será la dependencia lineal.

Además, el cálculo de la matriz de correlación puede suponer un punto de partida en las técnicas de reducción de la dimensionalidad, como en el análisis de componentes principales (ACP). Si se desea calcular la matriz de correlación del conjunto de datos `TIC2021`, que presenta las estadísticas de uso de las TIC en la Unión Europea 2021, para llevar a cabo posteriormente un ACP (ver Capítulo XX) se puede utilizar el paquete `corrplot`, que proporciona una forma elegante y versátil de representar la matriz de correlaciones[4].

```
library(corrplot)
mcor_tic <- cor(TIC2021)
```

[4]Una gran cantidad de ejemplos puede verse ejecutado `example(corrplot)`.

```
corrplot.mixed(mcor_tic, order = "AOE")
```

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

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

Por ejemplo, antes de llevar un ajuste lineal o de otro tipo con los datos `airquality` tal y como se hará en los capítulos XX, se podría hacer un AED previo entre las variables `Ozone` y `Temp` con el gráfico de la Figura 10.12.

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

Un caso particular es cuando la variable explicativa es el tiempo. En este caso estaremos hablando de series temporales, y la representación con líneas es más adecuada (véase la Figura 10.13).

Figura 10.12: Gráfico de dispersión del Ozono frente a la Temperatura

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

Tipos de Datos	Amplitud de los datos	Volumen	Frecuencia Actualización	Número de Columnas	Ejemplos	Propietarios o Stewards	Sistema Típico de Registro
Metadatos	Organización	Muy baja	Casi estática	Varias	Definiciones de atributos	Organización	Repositorio de Metadatos
Datos de Referencia Core	Organización, o casi toda la organización	Baja	Muy Baja	Pocas	SIC, Países, Códigos de Divisas, Esquemas ISSO	Normalmente externos a la organización	Repositorio de datos de referencia corporativo
Datos Maestros Comunes	Múltiples sistemas abarcando la organización	Medio	Bajo	Muchas	Cliente, Empleado, Producto, Cuentas Bancarias	Principalmente usuarios que lo crean o lo necesitan	Varios escenarios
Datos Transaccionales	Transacciones actuales	Alto	Medio	Muchas	Compras, inversiones, ...	Sistema que generaron las transiciones	Sistema que generaron las transiciones

Figura 10.13: Concentración media semanal de NOx en las estaciones de medición de Madrid (enero 2011- marzo 2022)

10.3.3. Variables cualitativas y cuantitativas

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

- Una variable numérica y una variable categórica: Gráficos de cajas o de violín para cada posible valor de la categórica (Figura 10.14), o bien gráficos de densidad para cada categoría (Figura 10.15).

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

Figura 10.14: Comparación de los niveles de PM10 en las Zonas de la ciudad de Madrid a efectos de Calidad del Aire durante la Calima de marzo de 2022

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

- Dos variables numéricas y una variable categórica: Gráfico de dispersión de las numéricas, y mapeado del color por la categórica.

Figura 10.15: Comparación de densidades de NOx por tipo de estación de medición

```
# periodo del estado de alarma
pm10_nox_mad <- contam_mad |>
  na.omit() |>
  filter(nom_abv %in% c("PM10", "NOx")) |>
  filter(between(fecha, left = as.Date("2020-03-14"), right =
  ↪  as.Date("2020-06-30"))) |>
  select(estaciones, zona, tipo, nom_abv, daily_mean, fecha) |>
  tidyr::pivot_wider(names_from = "nom_abv", values_from = "daily_mean",
↪  values_fn = mean)
```

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

- Más de dos variables numéricas y más de una categórica: gráfico de dispersión y mapeado de las otras variables a otros *aesthetics*. Combinación de geometrías y *aesthetics* (por ejemplo, añadir puntos con efecto *jitter* a un gráfico de cajas). La función `ggpairs` puede incluir también variables categóricas y mapearlas por ejemplo al color, consiguiendo una visualización muy rica.

En todos los casos anteriores, se pueden crear "facetas" para hacer un gráfico por cada combinación de variables categóricas, de forma que tengamos un buen número de variables representadas en un mismo "lienzo", como en la Figura 10.17.

Figura 10.16: Grafíco de dipersión entre las variables, 'NOx', PM110, 'zona' y 'tipo' durante el estado de alarma en la ciudad de Madrid

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

Figura 10.17: Grafíco de dipersión entre las variables, 'NOx', PM110, 'zona' y 'tipo' por estación de monitoreo durante el estado de alarma en la ciudad de Madrid

RESUMEN

Análisis exploratorio de una característica

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

Análisis exploratorio de varias características

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

Capítulo 11

Feature selection and engineering

Jorge Velasco López

11.1. Introducción

Como se indicaba en el capítulo anterior, la preparación de datos en un contexto de un poryecto de modelado predictivo, consiste en la transformación de datos sin procesar en una forma más adecuada para el modelado. Esta preparación puede ser un proceso minuciosamente laborioso e incluye tareas como la integración y limpieza de datos, como se vio en el capítulo anterior. En este capítulo se verán el resto de tareas de preprocesamiento:

- Selección de variables (*feature selection*): identificar aquellas variables de entrada que sean más relevantes para las fases posteriores del proceso de modelado.
- Transformaciones de la escala o distribución de la variable objetivo.
- Transformación de variables (*feature engineering*): derivación de nuevas variables a partir de los datos disponibles.
- Reducción de dimensionalidad: Creación de proyecciones compactas de los datos.
- Otro tipo de tratamientos, como el *split* de datos o el manejo de datos no balanceados.

La creación de variables predictoras a partir de los datos en bruto tiene una componente creativa. Requiere además de herramientas adecuadas y de experiencia para encontrar las mejores representaciones, apoyándose en lo posible en el conocimiento que se tenga de los datos durante la exploración de los mismos. Para seleccionar y configurar la preparación de datos, pueden usarse **estadísticas descriptivas**, para determinar si las operaciones de escalado pueden ser apropiadas. Las pruebas de hipótesis estadísticas se pueden utilizar para determinar si una variable coincide con una distribución de probabilidad dada. Se pueden usar diagramas para determinar si las variables están relacionadas y si lo están, en qué medida, proporcionando información sobre si una o más variables son redundantes o irrelevantes a la variable objetivo. Tambiés es posible apoyarse en visualizaciones, como **gráficas de los datos** para identificar si

una variable tiene valores atípicos y proporcionar información sobre la distribución de probabilidad que subyace a los datos y así poder decidir si una transformación de la distribución de probabilidad de una variable sería apropiada.

La fase posterior de modelización, también puede informar sobre la selección y configuración de los métodos de preparación de datos. Por ejemplo, la elección de **algoritmos** puede imponer requisitos y expectativas sobre el tipo y forma de variables de entrada en los datos (Boehmke and Greenwell, 2019). Así, podría ser necesario que las variables tengan una distribución de probabilidad específica, la eliminación de variables de entrada correlacionadas y/o la eliminación de variables que no estén fuertemente relacionadas con la variable objetivo.

Incluso preparando los datos para cumplir con las expectativas del modelo, es posible que no se obtenga todo su potencial, dado que hay representaciones de datos que son mejores que otras. El *feature engineering* ayudará a aumentar la eficacia de un modelo. Nótese que, para un conjunto determinado de datos y un problema concreto de modelado predictivo de clasificación o regresión, puede ser la primera vez que se realice una modelización y por tanto, sea necesario trabajar los datos de manera innovadora. Tanto es así, que se suele decir que se invierte hasta el 80 % del tiempo de análisis de datos en el proceso de preparación de datos.

El paquete `caret` (Kuhn, 2008) proporciona una interfaz unificada que simplifica el proceso de modelado empleando la mayoría de los métodos de aprendizaje estadístico implementados en **R**. Se puede decir que caret es un *meta-engine* (agregador) que permite aplicar casi cualquier algoritmo y se ha elegido como herramienta principal para la parte de preprocesamiento por su amplia difusión y por coherencia en la parte de *machine learning* del presente libro, que también lo utiliza. No obstante, se podrían usar otros, como el paquete `recipes` incluido en `tidymodels`. Los *meta-engines* y `caret` en particular brindan más coherencia en la forma en que especifican las entradas y se extraen las salidas, pero pueden ser menos flexibles que los algoritmos directos. En relación al preprocesamiento de los datos, además de una serie de funciones auxiliares, como `dummyVars()` y `rfe()`, incluye la función principal `preProcess()`, que ya se introdujo en el capítulo anterior y para la que el parámetro `method` permite establecer una lista de procesamientos:

- **Imputación**: `knnImpute`, `bagImpute` o `medianImpute`, para imputar valores missing como se vio en el capítulo anterior.
- **Creación y transformación de variables explicativas**: `center` (resta la media de los valores), `scale` (divide los valores por la desviación estándar), `range` (normaliza los valores), `BoxCox` (aplica una transformada de Box-Cox siendo los valores positivos), `YeoJohnson` (aplica una transformada Yeo-Johnson), `expoTrans` (aplica una transformación de potencia), `spatialSign` (transforma los datos a un círculo unitario de `p` dimensiones).
- **Selección de predictores y extracción de componentes**: `corr` (correlación), `nzv` (eliminar atributos con una varianza cercana a cero), `zv` (eliminar atributos con varianza cero), `pca` (transformar datos a los componentes principales), `ica` (transformar datos a los componentes independientes).

Es más probable que estas transformaciones sean útiles para algoritmos como el de regresión, métodos basados en instancias (también llamado *memory-based learning* como `K` vecinos

más cercanos-KNN- y aprendizaje de cuantificación vectorial-LVQ-), máquinas de vectores de soporte-SVM- y redes neuronales-NN- y menos probable que sean útiles para métodos basados en árboles y reglas. En `caret`, estas transformaciones pueden ser utilizadas de dos maneras:

1. **Independiente**: las transformaciones se pueden modelar a partir de datos de entrenamiento y aplicarse a múltiples conjuntos de datos. El modelo de la transformación se prepara utilizando la función `preProcess()` y se aplica a un conjunto de datos utilizando la función `predict()`.

2. **Entrenamiento**: las transformaciones se pueden preparar y aplicar automáticamente durante la evaluación del modelo. Las transformaciones aplicadas durante el entrenamiento se preparan usando `preProcess()` y se pasan a la función `train()` a través del argumento `preProcess`. En esta sección se presentan varios ejemplos de preprocesamiento de datos usando ambos métodos.

Se usará el dataset de `Madrid_Sale` (disponibles en el paquete de **R** `Idealista18`), con datos inmobiliaros del año 2018 y los datos de la tienda de comercio electrónico "Beauty eSheep" que se usa en los capítulos de *machine learning* y que se introdujo en el capítulo anterior. Las librerías que se usan, además de las menciondas de `tidyverse`, `idealista18` y `caret`, son `FSelector` (para la selección de atributos mediante métodos *embedded*), `rsample` (para realizar muestreo), `gridExtra` (para visualizar gráficos) y `corrplot` (para visualizar correlaciones).

11.2. *Feature Selection* (Selección de variables)

La selección de variables es el conjunto de técnicas para seleccionar el subconjunto de variables de entrada que sea más relevante para la modelización de la predicción de variable objetivo. Esto es importante porque: (i) variables redundantes de entrada pueden distraer o engañar a los algoritmos de aprendizaje, lo que posiblemente signifique un menor rendimiento predictivo; (ii) es deseable desarrollar modelos utilizando únicamente los datos que se requieren para hacer una predicción, tanto por el coste computacional como por la interpretabilidad del modelo utilizado.

Las técnicas de selección generalmente pueden agruparse en las que usan la variable de destino (supervisados) y los que no (no supervisados). Debido a la complejidad de la cuestión, se va a revisar únicamente las técnias supervisadas más relevantes, que se pueden dividir en:

- los de tipo filtro, que puntúan cada variable de entrada y permiten seleccionar un subconjunto (Brownlee, 2020);

- los métodos *wrapper*, que eligen las variables que dan como resultado el modelo de mejor rendimiento y

- los modelos intrínsecos o *embedded*, que seleccionan variables automáticamente como parte del ajuste del modelo durante el entrenamiento (como algunos modelos de regresión penalizados como `Lasso` y árboles de decisión y *random forests*).

La selección de variables también está relacionada con las técnicas de reducción de dimensionalidad, ya que en ambos métodos se busca reducir el número de variables de entrada para un modelo predictivo. La diferencia es que la reducción de la dimensionalidad crea una proyección de los datos que dan como resultado variables de entrada completamente nuevas. Así, la reducción de la dimensionalidad es una alternativa a la selección de variables.

11.2.1. Métodos de selección tipo Filtro

Los **métodos de selección de variables de filtro** usan técnicas estadísticas para evaluar la relación entre cada variable de entrada (también llamadas predictoras) y la variable de destino (también llamada objetivo o de salida). Las puntuaciones obtenidas se utilizan como base para clasificar y elegir las variables de entrada que se utilizarán en el modelo.

La elección de las técnicas estadísticas depende de los tipos de datos de las variables. Las medidas estadísticas utilizadas en la selección de variables basadas en filtros generalmente calculan una variable de entrada a la vez con la variable de destino. Como tales, se les conoce como medidas estadísticas univariantes.

En función de si la entrada/salida es numérica/categórica, se estaría tratando de un problema de modelado predictivo o de clasificación y las técnicas de selección de filtro serían diferentes. Por ejemplo, si las variables de entrada y salida fueran numéricas, sería un problema de modelado predictivo, para las que se suelen usar técnicas de coeficientes de correlación, como el de *Pearson* para una correlación lineal o métodos basados en rangos para una correlación no lineal (por ejemplo con el Coeficiente de rango de *Spearman*) y el método de Información Mutua. Si ambas fueran categóricas podría usarse la medida de correlación de chi-cuadrado. Sin embargo, no es habitual tener un conjunto de datos con solo un tipo de variable de entrada. Un enfoque para manejar diferentes tipos de datos de variables de entrada es seleccionar por separado variables de entrada numéricas y variables de entrada categóricas usando las métricas apropiadas.

Uno de los aspectos fundamentales en la selección de variables es comprobar si su varianza es cero o cercana a cero. Si la varianza es cercana a cero, significa que casi todas las observaciones tienen valores similares y por tanto, esas variables podrían ser descartadas, puesto que es muy probable que solo añadan ruido al modelo. Comprobar con `caret` si las observaciones tienen **varianza** cero se hace con la función `nearZeroVar()`. A continuación se muestra su uso con un ejemplo del dataset `Madrid_Sale`, incluido en el paquete de **R** `Idealista18`.

En el dataset se tienen varias variables numéricas (como las correspondientes al número de baños, metros cuadrados, precio, latitud, longitud, si tiene terraza, etc.). Para comprobar si tienen o no varianza cero se puede ejecutar el siguiente código, que devuelve entre otras la variable `nzv=FALSE` (*near zero variance*) para casi todas las variables, excepto `PARKINGSPACEPRICE, ISDUPLEX, ISSTUDIO, ISINTOPFLOORyBUILTTYPEID_1` que podrían descartarse para el modelo:

```
numeric_cols <- sapply(Madrid_Sale, is.numeric)
variance <- nearZeroVar(Madrid_Sale[numeric_cols], saveMetrics = T)
# Con el argumento saveMetrics, se guardan los valores que se han utilizado
↪  para los cálculos.
```

```
# Se muestran los primeros resultados
head(variance, n = 2)
#>        freqRatio  percentUnique   zeroVar   nzv
#> PERIOD  2.019617   0.004218742   FALSE   FALSE
#> PRICE   1.076923   2.911986500   FALSE   FALSE
```

Otra de las cuestiones importantes, es la **correlación entre variables**. Existen varios modelos, como la regresión lineal y la regresión logística para las que una de las bases del modelo es la no colinealidad o multicolinealidad, por lo que no deberían haber variables correlacionadas.

Para comprobarla, se puede usar la función `findCorrelation()` de `caret`a la que se le pasa una matriz de correlaciones, con lo que se obtiene qué variables habría que eliminar en caso necesario. Con el paquete de `corrplot` se genera la visualización de la Fig. 11.1:

Figura 11.1: Correlación

```
madrid_cor <- cor(Madrid_Sale[numeric_cols])
findCorrelation(madrid_cor)
# Se visualiza
correlationMatrix <- cor(madrid_cor[, 1:5])
corrplot(correlationMatrix, method = "circle")
```

En este caso, no habría variables altamente correlacionadas, aunque sí habría que analizar principalmente la relación entre `CONSTRUCTEDAREA` y `BATHNUMBER`. Si las hubiera, habría que considerar eliminar determinadas variables. De manera similar, habría que controlar que no haya combinaciones lineales, para lo que se puede usar la función `findLinearCombos()`.

La colinealidad ocurre cuando los predictores del modelo están relacionados de tal manera que constituyen una combinación lineal entre sí. Aunque no empeoraría el poder predictivo del modelo, sí dificultaría la evaluación de la importancia de los predictores individuales.

Las variables perfectamente correlacionadas son redundantes y es posible que no agreguen valor al modelo y, si se eliminan, serían más rápidas de entrenar. Sin embargo, dos variables que no están fuertemente correlacionadas aún podrían ser importantes para el modelo. En caso de duda, lo más seguro sería entrenar el modelo y observar su rendimiento.

11.2.2. Métodos de selección de variables tipo *wrapper*

Otro enfoque consiste en utilizar un método contenedor o *wrapper* (Saeys et al., 2007), que realiza una búsqueda a través de diferentes combinaciones o subconjuntos de variables de entrada para comprobar el efecto que tienen en la precisión del modelo.

Hay varias alternativas:

- Evaluar las variables individualmente y seleccionar las n principales variables que obtienen una buena precisión. Sin embargo, al probar el modelo repetidamente con solo una de las variables a la vez, se pierde la información de las dependencias entre variables.
- Observar la precisión del modelo para todas las combinaciones de variables posibles. En este sentido, se puede utilizar un algoritmo de búsqueda global estocástica, como los algoritmos genéticos. Aunque efectivos, estos enfoques pueden ser computacionalmente muy costosos, especialmente para grandes conjuntos de datos de entrenamiento y modelos más sofisticados.
- La selección *forward* es un algoritmo simple en el que comienza con una sola variable y agrega secuencialmente la siguiente variable, actualizando el modelo y observando su rendimiento.
- La selección *backward* es el enfoque inverso de la selección *forward*. Se comienza con todas las variables y luego se eliminan iterativamente la variable de menor rendimiento, mientras se actualiza el modelo. Esto ayuda a comprender si la última variable clasificada era realmente útil.
- Un método automático popular para la selección de variables proporcionado por el paquete `caret`se llama "Eliminación de variables recursivas" o RFE.

El siguiente ejemplo ilustra la última de las alternativas, el **método RFE**, en el conjunto de las variables numéricas del `data.frame` de `Madrid_Sale`. Se utiliza un algoritmo *Random Forest* en cada iteración para evaluar el modelo. El algoritmo está configurado para explorar todos los subconjuntos posibles de los atributos seleccionados. En la Fig. 11.2, se puede comprobar que en 3 atributos se estabiliza el RMSE.

```
# Se toma una muestra con el paquete rsample y se seleccionan las variables de
↪  interés
Madrid_Sale_num <- Madrid_Sale[, 3:9] |>
  dplyr::select(-UNITPRICE)
```

```
Madrid_Sale_num_sample <- sample(1:nrow(Madrid_Sale_num), size = 5000, replace
↪  = FALSE)
Madrid_Sale_num_sample <- Madrid_Sale_num[Madrid_Sale_num_sample, ]
# Para asegurar que los resultados sean repetibles
set.seed(7)
# Definir el control usando una función de selección de random forest
control <- rfeControl(functions = rfFuncs, method = "cv", number = 10)
# Ejecutar el algoritmo RFE sobre el dataset de Sacramento de solo variables
↪  numéricas
results <- rfe(Madrid_Sale_num_sample[, 2:6], Madrid_Sale_num_sample[, 1],
↪  sizes = c(1:5), rfeControl = control)
# sumarizar los resultados
print(results)
# listar las variables resultantes
predictors(results)
# Graficar los resultados
plot(results, type = c("g", "o"))
```

Figura 11.2: Selección RFE

11.2.3. Métodos de selección tipo Embedded

Finalmente, hay algunos algoritmos de aprendizaje automático que realizan la selección automática de funciones como parte del aprendizaje del modelo. Se podría aludir a estas técnicas como **métodos de selección de variables intrínsecas (o *embedded*)**. Esto incluye algoritmos como modelos de regresión penalizados como Lasso y árboles de decisión, incluidos los *random forests*.

El siguiente ejemplo carga el conjunto de datos de `Madrid_Sale` con una selección de variables numéricas (`PRICE`, `CONSTRUCTEDAREA`, `ROOMNUMBER`, `BATHNUMBER`, `HASTERRACE`, `HASLIFT`)

y construye un modelo de aprendizaje de cuantificación vectorial o LVQ, (Kohonen, 1995). Luego, se usa la función `varImp()` para estimar la importancia de la variable, que se muestra en consola y se representa gráficamente. En la Fig. 11.3 se muestra que entre los atributos analizados, los de `CONSTRUCTEDAREA, BATHNUMBER y ROOMNUMBER` son los 3 más importantes del conjunto de datos para todas las categorías y `HASTERRACE` es el menos importante, para la variable objetivo `PRICE`, transformada como se explicará en los siguientes apartados mediante *binning*.

```
# Para asegurar que los resultados sean repetibles
set.seed(7)
# Preparar el esquema de entrenamiento en este caso "repeated CV"
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
# Realizar el *Binning* con un número determinado de bins
df3 <- Madrid_Sale_num_sample |>
  dplyr::mutate(price_bin = cut(PRICE, breaks = c(0, 250000, 500000, 750000,
↪  10000000), labels = c("0k_250k", "250k_500k", "500k_750k", ">750k"),
↪  include.lowest = TRUE)) |>
  dplyr::select(-PRICE)
# Entrenar el modelo:
model <- train(price_bin ~ ., data = df3, method = "lvq", preProcess =
↪  "scale", trControl = control)
importancia <- varImp(model, scale = FALSE) # Estimar la importancia de la
↪  variable
plot(importancia) # Graficar la importancia
```

Figura 11.3: Método Embedded

11.3. Transformaciones de escala y de la distribución de la variable objetivo

11.3.1. Transformaciones de la variable objetivo

Aunque no siempre es necesario, la transformación de la variable objetivo puede llevar a una mejora predictiva, especialmente con modelos paramétricos. Por ejemplo, los modelos de regresión lineal ordinarios asumen que los errores de predicción (y por lo tanto la respuesta) se distribuyen normalmente. Pero puede ocurrir que la variable objetivo tenga valores atípicos y la suposición de normalidad no se cumpla.

Para minimizar la asimetría de la variable respuesta, se puede usar una transformación log (u otra). Esta sería una alternativa al uso de la función de pérdida del "error logarítmico cuadrático medio" (RMSLE) como medida de evaluación del modelo.

Existen dos enfoques principales para ayudar a corregir las variables objetivo con sesgo positivo:

- Normalizar con una transformación log. Esto transformará la mayoría de las distribuciones sesgadas a la derecha para que sean aproximadamente normales. Sin embargo, hay que considerar el preprocesamiento como la creación de un flujo de trabajo que pueda revisarse y volver a usarse. Para esto, se puede usar el paquete `recipes` o la función `caret::preProcess()` [1]. En la Fig. 11.4, se puede ver que tomar logaritmos da buen resultado para normalizar esta distribución.

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

- Como segunda opción se puede usar una transformación de Box-Cox, que es más flexible que la transformación logarítmica y se puede encontrar la función adecuada a partir de una familia de transformadas de potencia, que transformarán la variable a lo mas parecido a una distribución normal (Sakia, 1992) [2].

```
respuesta_boxcox <- preProcess(Madrid_Sale_num_sample, method = "BoxCox")
trainBC <- predict(respuesta_boxcox, Madrid_Sale_num_sample)
respuesta_boxcox
#> Created from 5000 samples and 2 variables
#>
```

[1] Si la variable objetivo tiene valores negativos o ceros, una transformación logarítmica producirá NaN e $-Inf$, respectivamente. Si los valores de respuesta no positivos son pequeños (por ejemplo, entre $-0{,}99$ y 0), se puede aplicar una pequeña compensación como con la función `log1p()` que agrega 1 al valor antes de aplicar una transformación de registro. Si los datos consisten en valores ≥ -1, se debe usar la transformación de Yeo-Johnson. Por tanto, si se usa una transformación log o Box-Cox, no se deben centrar los datos primero ni realizar ninguna operación que pueda hacer que los datos no sean positivos.

[2] En el centro de la transformación de Box-Cox hay un exponente, lambda (λ), que varía de -5 a 5. Se consideran todos los valores de λ y se estima el valor óptimo para los datos dados a partir de los datos de entrenamiento. El "valor óptimo" es el que resulta en la mejor transformación a una distribución normal aproximada.

Figura 11.4: Variable respuesta log

```
#> Pre-processing:
#> - Box-Cox transformation (2)
#> - ignored (0)
#>
#> Lambda estimates for Box-Cox transformation:
#> -0.3, -0.3
```

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

NOTA
El paquete `recipes`, incluido en `tidymodels`, permite desarrollar el modelo de transformación de variables de forma secuencial. La idea detrás del paquete es similar a `caret::preProcess()`, donde se quiere crear el modelo de preprocesamiento pero aplicarlo más tarde y dentro de cada remuestreo. Hay tres pasos principales para crear y aplicar la transformación de variables con recipes, *recipe* (donde se definen los pasos para crear el plan), *prep* (donde se estiman los parámetros en función de los datos de entrenamiento) y *bake* (donde se aplica el modelo a nuevos datos). Sin embargo, a diferencia de `caret`, no manejan automáticamente las variables categóricas y se requiere crear variables ficticias manualmente.

11.3.2. Escalado de datos

También hay que considerar la escala en la que se miden las variables individuales. Los modelos que incorporan funciones lineales en las variables de entrada, son sensibles a la escala de esas

variables. Muchos algoritmos usan estas funciones, algunas de manera más obvia (como Modelos Lineales Generalizados-GLM- y regresión regularizada) que otras (como redes neuronales-NN-, SVM y análisis de componentes principales). Otros ejemplos incluyen algoritmos que utilizan medidas de distancia, como la distancia euclidia (p. ej., KNN, agrupación de k-medias y agrupación jerárquica). Para estos modelos, a menudo es aconsejable estandarizar las variables de entrada. Las funciones de estandarización incluyen el centrado y la escala para que las variables numéricas tengan una media cero y una varianza unitaria, lo que proporciona una unidad de medida comparable común en todas las variables.

La función `preProcess()` de `caret` acepta dos tipos principales: `center`, que resta el promedio a los valores (todos tendrán promedio 0); `scale`, que divide los valores entre la desviación estándar (todos tendrán desviación típica 1) y `range`, que normaliza los datos, haciendo que éstos tengan un rango de 0 a 1.

```
preprocess <- preProcess(Madrid_Sale_num, method = "center")
head(predict(preprocess, Madrid_Sale_num)[1:3], n = 3)
#>       PRICE    CONSTRUCTEDAREA     ROOMNUMBER
#> 1 -270110.11       -54.39716       -1.5808996
#> 2 -161110.11       -47.39716       -1.5808996
#> 3  -23110.11       -26.39716       -0.5808996
```

Con esta función y para este caso, `caret` centra los datos de 6 variables, correspondientes a las variables numéricas seleccionadas. La función `preProcess()` no está pensada para transformar los datos en el momento, sino para hacer la transformación en el proceso de entrenamiento o de predicción. Para ello, en este caso se pasa el preprocesamiento y los datos a la función `predict()` devolviendo caret todos los datos con el procesamiento ya aplicado (en este caso, habiendo restado la media). La combinación de las transformaciones de scale y center puede servir para estandarizar los datos. Los atributos tendrán un valor medio de 0 y una desviación estándar de 1 denominándose estandarización **z-score**. Para ello, se usaría la sentencia `method=c(çenter", "scale")`. Otra estandarización ampliamente usada es la **min-max**.

Cuando se estandariza, hay que hacerlo tanto en los datos de entrenamiento (*train*) como en los de prueba (*test*), para que se basen en la misma media y varianza. En todo caso hay que tener en cuenta que nunca deben de usarse los datos que se han apartado para el conjunto de *test* en el proceso de estandarización.

11.4. *Feature engineering*

La mayoría de los modelos requieren que los predictores tomen forma numérica, con alguna excepción. Por ejemplo, los modelos basados en árboles manejan de manera natural variables numéricas o categóricas. Pero incluso estos modelos pueden mejorarse con el uso del preprocesamiento de variables categóricas. Las siguientes secciones se centran en algunos de los enfoques más comunes para diseñar variables categóricas.

Estos métodos generan nuevas variables, lo que aumenta la precisión del modelo y las predicciones generales. Son de dos tipos, de Agrupamiento o *Binning*, donde se crean agrupaciones

(o *bins*) para variables continuas y de Codificación, donde las variables numéricas se forman a partir de variables categóricas.

11.4.1. *Binning*

Binning es el agrupamiento que se realiza para crear agrupamientos de variables continuas convirtiéndolas en categóricas. Hay dos tipos de agrupamiento, 1) no supervisado (agrupamiento automático o manual) y 2) supervisado, que implica la creación de *bins* para la variable continua mientras se tiene en cuenta también la variable objetivo. Sin embargo, el *Binning* debe usarse con moderación, ya que puede haber una pérdida en el rendimiento del modelo (Kuhn et al., 2013). En el ejemplo que se mostraba en apartados anteriores, se hacía *Binning* sobre la variable de `PRICE` para emplear un método de selección de variables tipo *embedded*. A continuación, se muestra una forma alternativa de hacer el *bin*.

```
df2 <- Madrid_Sale_num_sample |>
  mutate(price_bin = cut(PRICE, breaks = 4)) |>
  dplyr::select(-PRICE)
head(df2, n = 2)
#>       CONSTRUCTEDAREA ROOMNUMBER BATHNUMBER HASTERRACE HASLIFT
#> 72912              80          3          1          1       0
#> 64401             121          3          2          0       1
#>                 price_bin
#> 72912 (2.69e+04,2.06e+06]
#> 64401 (2.69e+04,2.06e+06]
```

11.4.2. Codificación

Muchos modelos requieren que todas las variables predictoras sean numéricas. En consecuencia, se puede necesitar transformar las variables categóricas en representaciones numéricas para que estos algoritmos puedan procesarse. La codificación es el proceso en el que se crean variables numéricas a partir de variables categóricas. Es de dos tipos, 1) **Codificación de etiquetas** (implica asignar a cada etiqueta un entero o valor único según el orden alfabético, que es la codificación más popular y ampliamente utilizada) y 2) **Codificación *One-hot*** (implica la creación de variables adicionales o *dummies* sobre la base de valores únicos para cada categoría de la variable categórica). Esta última se suele usar cuando se agrupan categorías que ocurren con poca frecuencia y en general, muchos algoritmos trabajan sobre ellas de manera eficiente. Para crear variables *dummies* con `caret`, se puede usar la función `dummyVars()` y aplicar un `predict()` para obtener los datos resultantes. La estrategia que implementa es crear una columna para cada valor distinto que exista en la variable que se está codificando y, para cada registro, marca con un 1 la columna a la que pertenezca dicho registro y deja las demás con 0. Para el dataset de la tienda de comercio electrónico "Beauty eSheep", se haría lo siguiente.

```
# Se obtiene el modelo de *one hot encoding*
# se aplica el modelo para obtener un data.frame con las variables ya
↪  modificadas
# se debe hacer lo mismo con los de test.
dp_dummies <- dummyVars(" ~ .", data = df_pre_dummies)
df_pos_dummies <- predict(dp_dummies, newdata = dp_entr)
```

Adicionalmente, en este mismo ejemplo, se crea un `data.frame` uniendo estas variables `df_pos_dummies` con el resto de variables del modelo:

```
# Se obtiene la tabla con el resto de variables que no se han transformado:
↪  las variables que son numéricas y la clase objetivo
df_pos_resto <- dp_entr |>
  select(-starts_with(c("des_", "ind_")))
# Se obtiene así el data.frame todas las variables numéricas: las columnas de
↪  indicadores, las columnas dummies y el resto de columnas no transformadas
↪  (la clase objetivo queda al final siendo un factor):
dp_entr_num <- df_pos_ind |>
  bind_cols(df_pos_dummies, df_pos_resto)
```

11.5. Reducción de dimensionalidad

La reducción de dimensionalidad es un enfoque alternativo para filtrar las variables no informativas sin eliminarlas mediante el *feature selection*, que generalmente se usa para variables numéricas. Como se explica en el capítulo de ***ref(ACP)***, el espacio de un conjunto de variables podría reducirse proyectándolo a un subespacio de variables de menor dimensión utilizando componentes principales.

Para aplicar un análisis de componentes principales (PCA por sus siglas en inglés) en **R** con `caret`, se indica el valor `pca` al parámetro method de la función `preProcess()`. Asimismo, con el parámetro `thresh` se puede indicar el porcentaje de la variabilidad deseado. Igualmente, se podrían combinar métodos, por ejemplo haciendo `method=c(çenter", "scale", "pca")`.

```
pre_pca <- preProcess(Madrid_Sale_num, method = "pca", thresh = 0.8)
predict(pre_pca, Madrid_Sale_num)
```

11.6. Otras transformaciones

11.6.1. Particionado de datos

Un objetivo principal del proceso de aprendizaje automático es encontrar el algoritmo que prediga con mayor precisión los valores futuros en función de un conjunto de variables. En

otras palabras, un algoritmo que, no solo se ajuste bien a nuestros datos pasados sino, lo que es más importante, que prediga un resultado futuro con precisión. Esto se llama la generalización del algoritmo.

Para proporcionar una comprensión precisa de la generalización del modelo óptimo final, se pueden dividir los datos en conjuntos de datos de entrenamiento y prueba:

- **Conjunto de entrenamiento (*train*)**: estos datos se utilizan para desarrollar conjuntos de funciones, entrenar los algoritmos, ajustar hiperparámetros, comparar modelos y todas las demás actividades necesarias para elegir un modelo final.
- **Conjunto de prueba (*test*)**: habiendo elegido un modelo final, estos datos se utilizan para estimar una evaluación imparcial del rendimiento del modelo, que se puede denominar" error de generalización".

Dada una cantidad fija de datos, las recomendaciones típicas para dividir los datos en divisiones de prueba de entrenamiento incluyen 60 % (entrenamiento)–40 % (`test`), 70 %–30 % u 80 %–20 %. Las dos formas más comunes de dividir los datos incluyen el muestreo aleatorio simple y el muestreo estratificado, que explican en detalle en capítulos dedicados de este libro.

11.6.1.1. Muestreo aleatorio simple

La forma más sencilla de dividir los datos en conjuntos de entrenamiento y `test` es tomar una muestra aleatoria simple. Esto no controla ningún atributo de datos, como la distribución de su variable objetivo (Y). Hay varias formas de dividir nuestros datos en **R**. Por ejemplo, con `caret` se haría lo siguiente para producir una división de 70 a 30 en los datos:

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

11.6.1.2. Muestreo estratificado

Si se desea controlar el muestreo para que los conjuntos de entrenamiento y prueba tengan similares distribuciones, se puede usar muestreo estratificado. Esto es habitual en problemas de clasificación donde la variable objetivo puede estar notablemente desbalanceada (por ejemplo, 90 % de las observaciones con respuesta "Sí" y 10 % con respuesta "No"). Sin embargo, también se puede aplicar el muestreo estratificado a problemas de regresión para conjuntos de datos que tienen un tamaño de muestra pequeño y donde la variable objetivo se desvía mucho de la normalidad.

La forma más sencilla de realizar un muestreo estratificado en una variable objetivo es usar el paquete `rsample`, donde se especifica la variable objetivo para estratificar. Lo siguiente ilustra que en nuestros datos originales la variable del precio con *binning* (`price_bin`) tiene una respuesta desbalanceada. Al aplicar el muestreo estratificado, tanto los conjuntos de entrenamiento como de prueba tienen distribuciones de respuesta aproximadamente iguales.

```
table(df3$price_bin) |> prop.table()
#         0k_250k 250k_500k 500k_750k     >750k
#         0.4776    0.2914    0.1132    0.1178
split_strat <- initial_split(df3, prop = 0.7, strata = "price_bin")
train_strat <- training(split_strat)
test_strat <- testing(split_strat)
# Tasa de respuesta consistente entre los datos train y test
table(train_strat$price_bin) |> prop.table()
#   0k_250k 250k_500k 500k_750k     >750k
#   0.4777015 0.2913093 0.1132075 0.1177816
table(test_strat$price_bin) |> prop.table()
#   0k_250k 250k_500k 500k_750k     >750k
#   0.4773635 0.2916112 0.1131824 0.1178429
```

Para el ejemplo que se presenta en este libro para el *machine learning* de la tienda de comercio electrónico "eSheep" se obtenía, después de un proceso de integración descrito en el capítulo anterior, el `data.frame` `dp_fac_cliente_venta`, a partir del cual se obtenía `dp_entr` (datos de entrenamiento de los modelos en su escala original) y `dp_test` (datos para prueba de los modelos en su escala original):

```
# Se parte en entrenamiento y test usando muestreo estratificado por la
↪  variable objetivo
trainIndex <- createDataPartition(dp_fac_cliente_venta_pro13[, dep], p = .8,
↪  list = F, times = 1)
dp_entr <- dp_fac_cliente_venta_pro13[trainIndex, ]
dp_test <- dp_fac_cliente_venta_pro13[-trainIndex, ]
# Una selección de variables numéricas da como resultado los data.frames
↪  dp_entr_num y dp_test_num, para los conjuntos de entrenamiento y test,
↪  respectivamente.
```

11.6.2. Técnicas para manejar datos no balanceados

Los datos utilizados en distintas áreas a menudo tienen menos del 1 % de eventos raros pero "interesantes" (por ejemplo, estafadores que usan tarjetas de crédito, usuarios que hacen clic en anuncios o servidores dañados que escanean su red). Sin embargo, la mayoría de los algoritmos de aprendizaje automático no funcionan bien con conjuntos de datos desbalanceados (Kuhn et al., 2013). Hay varias técnicas para lidiar con esto de las que a continuación se introducen algunas:

- ***Downsampling*** equilibra el conjunto de datos al reducir el tamaño de las clases abundantes para que coincidan con las frecuencias en la clase menos prevalente. Este método se utiliza cuando la cantidad de datos es suficiente. Manteniendo todas las muestras en la clase rara y seleccionando aleatoriamente un número igual de muestras en la clase abundante, se puede recuperar un nuevo conjunto de datos equilibrado para modelado adicional.
- ***Upsampling*** se usa cuando la cantidad de datos es insuficiente. Trata de equilibrar el conjunto de datos aumentando el tamaño de las muestras más raras. En lugar de deshacerse de las muestras abundantes, se generan nuevas muestras raras mediante la repetición o el *bootstrapping*.
- **Creación de datos sintéticos** (Chawla et al., 2002): Esta técnica consiste en balancear el conjunto de entrenamiento generando nuevos registros sintéticos, esto es, inventados de la clase minoritaria. Existen diversos algoritmos que realizan esta tarea siendo uno de los más conocidos la técnica de SMOTE (*Synthetic Minority Oversampling Technique*).
- **Otras técnicas**, como que el algoritmo implemente mecanismos para dar mayor peso a los casos de la clase minoritaria, etc

Por ejemplo, en capítulos posteriores se modelará por ejemplo el algoritmo **RPART** (algoritmos de árboles de regresión y clasificación) con *downsampling*, para obtener mejora en el rendimiento:

```
# Se especifica que el modelo se entrene con downsampling
md_conf_model$fitControl$sampling <- "down"
#  Se le indica que haga un centrado a los datos
md_conf_model$fitControl$preProcess <- "center"
```

Para todos los modelos de *machine learning* de capítulos posteriores, se hace previamente una parametrización común en esta fase de preparación de los datos y se guarda en la lista `md_conf_model`. Esta lista incluye los elementos `dep`, `form`, `fitControl`, `positive_class`, `metric` y `seed`, que se conforman así:

```
# nombre de la columna de la clase objetivo
dep <- colnames(dp_entr[ncol(dp_ENTR)])
# metrica de validacion
metric <- "ROC"
# preparar cv
fitControl <- trainControl(method = "cv", number = 10, savePredictions = T,
↪  summaryFunction = twoClassSummary, classProbs = T)
# clase positiva el primer nivel del factor
positive_class <- levels(dp_entr[, dep])[1]
# fórmula con la clase objetivo en función del resto
form <- formula(paste0(dep, "~."))
```

Hay que considerar, que no existe una ventaja absoluta de un método de muestreo sobre otro. La aplicación de estos métodos depende del caso de uso al que se aplica y del conjunto de datos en sí. La función de `caret` para implementar estas técnicas está en `?caret::trainControl()`), como se ve en el ejemplo anterior.

11.6.3. Métodos de remuestreo

En la sección anterior se indicaba que los datos deben dividirse en conjuntos de entrenamiento y prueba y que no debía usarse el conjunto de prueba para evaluar el rendimiento del modelo durante la fase de entrenamiento. Para evaluar el rendimiento del modelo, una opción sería evaluar una métrica de error basada en los datos de entrenamiento. Desafortunadamente, esto conduce a resultados sesgados, ya que algunos modelos pueden funcionar muy bien con los datos de entrenamiento pero no generalizarse bien a un nuevo conjunto de datos.

Un segundo método es utilizar un enfoque de validación, que implica dividir aún más el conjunto de entrenamiento para crear dos partes: un conjunto de entrenamiento y un conjunto de **validación.** Entonces se puede entrenar el modelo en el nuevo conjunto de entrenamiento y estimar el rendimiento en el conjunto de validación. Lamentablemente, la validación con un solo conjunto de reserva puede ser muy variable y poco confiable a menos que esté trabajando con conjuntos de datos muy grandes (Molinaro et al., 2005).

En el capítulo de "Técnicas de Modelización estadística avanzadas" se explicaban los métodos de remuestreo, que brindan un enfoque alternativo al permitir ajustar repetidamente un modelo a partir de los datos de entrenamiento y probar su rendimiento en otras partes. Los dos métodos de remuestreo más utilizados incluyen la validación cruzada de kfold y el *bootstrapping*.

11.6.3.1. Validación cruzada k-fold (*kfold cross validation*)

La validación cruzada kfold (kfold CV) es un método de remuestreo que divide aleatoriamente los datos de entrenamiento en k grupos (*folds*) de aproximadamente el mismo tamaño. El modelo se ajusta en k menos 1 grupos y el último se usa para calcular el rendimiento del modelo. Este procedimiento se repite k veces; cada vez, un grupo diferente se trata como el conjunto de validación. Este proceso da como resultado k estimaciones del error de generalización. Por lo tanto, la estimación del kfold CV se calcula promediando los errores de prueba k, lo que nos proporciona una aproximación del error de generalización que podríamos esperar en datos nuevos.

En consecuencia, con kfold CV, cada observación en los datos de entrenamiento se mantendrá una vez para ser incluida en el conjunto de prueba, como se ilustra en 11.5. En la práctica, normalmente se usa k=5 o k=10. No existe una regla formal en cuanto al tamaño de k; sin embargo, a medida que k aumenta, la diferencia entre el rendimiento estimado y el rendimiento real que se verá en el conjunto de prueba disminuirá. Por otro lado, el uso de k demasiado grande puede introducir cargas computacionales. Además, en (Molinaro et al., 2005) encontraron que k=10 funcionaba de manera similar a la validación cruzada dejando uno fuera (LOOCV), que es el enfoque más extremo (es decir, establecer k=n).

Aunque el uso de $k \geq 10$ ayuda a minimizar la variabilidad en el rendimiento estimado, el kfold CV tiende a tener una mayor variabilidad que el *bootstrapping* (que se analiza a continuación). (Kim, 2009) demostró que repetir kfold CV puede ayudar a aumentar la precisión del error de generalización estimado.

Para el caso de "Beauty eSheep", cuando se entrena el algoritmo se le pasa el parámetro de `trControl` entre los que está `fitControl`, que como se veía anteriormente está parametrizado

Figura 11.5: Remuestro k fold cv

para realizar un kfold cv de 10 *folds*.

11.6.3.2. *Bootstrapping*

Es una muestra aleatoria de los datos tomados con reemplazo (Efron and Tibshirani, 1986). Esto significa que, después de seleccionar un dato para incluirlo en el subconjunto, aún está disponible para una selección posterior. Una muestra bootstrap tiene el mismo tamaño que el conjunto de datos original a partir del cual se construyó. La Fig. 11.6 proporciona un esquema de muestreo bootstrap donde cada muestra contiene 12 observaciones al igual que en el conjunto de datos original. Además, el muestreo bootstrap contendrá aproximadamente la misma distribución de valores (representados por colores) que el conjunto de datos original.

Figura 11.6: Remuestro bootstrap

Dado que las observaciones se replican en *bootstrapping*, tiende a haber menor variabilidad en la medida del error en comparación con kfold CV. Sin embargo, también puede aumentar el sesgo de su estimación de error. Esto puede ser un problema con conjuntos de datos más pequeños. Sin embargo, para la mayoría de los conjuntos de datos de tipo medio o grande (por ejemplo, $n \geq 1000$), no ocasiona problemas. Se pueden crear muestras *bootstrap* fácilmente con `rsample::bootstraps()`, como se ilustra en el fragmento de código a continuación.

```
bootstraps(df3, times = 10)
```

Para series temporales, para incorporar el origen móvil y otros procedimientos de remuestreo de series temporales (Hyndman and Athanasopoulos, 2018) es el recurso más usado, centrado en **R**.

11.6.4. Ajuste de hiperparámetros

Aunque no se trata de transformación de preprocesamiento, sino ya de la fase de modelización, se introduce aquí el concepto de hiperparámetros (también conocidos como parámetros de ajuste), que sirven para para controlar la complejidad de los algoritmos de aprendizaje automático y, por tanto, la compensación entre sesgo y varianza. No todos los algoritmos tienen hiperparámetros (por ejemplo, mínimos cuadrados ordinarios-MCO-). Sin embargo, la mayoría tiene al menos uno. La configuración adecuada de estos hiperparámetros a menudo depende de los datos y el problema en cuestión y no siempre se puede ajustar solo con los datos de entrenamiento. En consecuencia, se requiere un método para identificar la configuración óptima.

Supongamos que los valores de k más pequeños (por ejemplo 2, 5 o 10) conducen a una varianza alta (pero un sesgo más bajo) y los valores más grandes (por ejemplo, 150) conducen a un sesgo alto (pero una varianza más baja). Entonces, el valor óptimo de k podría ser entre 20 y 50, pero hay que preguntarse por el valor óptimo. Una forma de realizar el ajuste de hiperparámetros es probar los hiperparámetros manualmente hasta que se encuentre la mejor combinación de valores de hiperparámetros que resulten en la mayor precisión predictiva (medida usando k fold CV, por ejemplo), aunque puede ser un trabajo muy tedioso dependiendo de la cantidad de hiperparámetros. Un enfoque alternativo es realizar una búsqueda en cuadrícula, que es un enfoque automatizado para buscar en muchas combinaciones de valores de hiperparámetros.

11.6.5. Evaluación de modelos

De manera similar al punto anterior, aunque no se trata de transformaciones de la fase de preprocesamiento, se introduce aquí el concepto de evaluación de modelos. Tradicionalmente, el rendimiento de los modelos estadísticos se basaba en gran medida en pruebas de bondad de ajuste y evaluación de residuos. Desafortunadamente, pueden derivarse conclusiones equívocas de los modelos predictivos que pasan este tipo de evaluaciones.

En la actualidad, para analizar el rendimiento del modelo, se suele evaluar la precisión predictiva a través de funciones de pérdida (*loss functions*). Las funciones de pérdida son métricas que comparan los valores predichos con el valor real. Al realizar métodos de remuestreo, se evalúan los valores pronosticados para un conjunto de validación en comparación con el valor objetivo real. Por ejemplo, en la regresión, una forma de medir el error es tomar la diferencia entre el valor real y el predicho para una observación determinada (esta es la definición habitual de residuo en la regresión lineal ordinaria). El error de validación general del modelo se calcula agregando los errores en todo el conjunto de datos de validación. Es importante considerar el contexto del problema al identificar la métrica de rendimiento a usar. Además, al comparar varios modelos, hay que hacerlo con la misma métrica. Para los **modelos de predicción** caben destacar las siguientes métricas, todos con objetivo de minimizar menos el R^2 que es de maximizar:

- El error cuadrático medio (**MSE**) es el promedio del error cuadrático, siendo ésta (junto a la siguiente) la métrica de error más común utilizada. Siendo x e y, D vectores dimensionales, y x_i el valor de la i dimensión de x, $MSE = \sum_{i=1}^{D}(x_i - y_i)^2$
- **RMSE**: Raíz del error cuadrático medio. La raíz se toma para que su error esté en las mismas unidades que su variable de respuesta. $RMSE = \sqrt{MSE}$

- **Desviación**: Abreviatura de desviación residual media. En esencia, proporciona un grado en el que un modelo explica la variación en un conjunto de datos cuando se utiliza la estimación de máxima verosimilitud.
- **MAE**: Error absoluto medio. Similar a MSE pero en lugar de elevar al cuadrado, solo toma la diferencia media absoluta entre los valores reales y predichos. $MAE = \sum_{i=1}^{D} |x_i - y_i|$
- **RMSLE**: Raíz del error logarítmico cuadrático medio. Similar a RMSE pero realiza un log() en los valores reales y predichos antes de calcular la diferencia. $RMSLE = \sqrt{(log(x_i + 1) - log(y_i + 1))^2)}$.
- R^2: Esta es una métrica popular que representa la proporción de la varianza en la variable dependiente que es predecible a partir de la(s) variable(s) independiente(s).

Para el caso de la **clasificación** destacan las siguientes medidas, todos con objetivo de minimizar:

- **Clasificación errónea**: tasa de clasificación incorrecta en porcentaje.
- **Error medio por clase**: tasa de error promedio para cada clase.
- **MSE**: Error cuadrático medio. Es la distancia de 1.0 a la probabilidad sugerida.
- **Entropía cruzada** (también conocida como pérdida de registro o desviación): similar a MSE pero incorpora un registro de la probabilidad predicha multiplicada por la clase real. En consecuencia, esta métrica penaliza las predicciones en las que se predice una pequeña probabilidad para la clase verdadera.
- **Índice de Gini**: se utiliza principalmente con métodos basados en árboles y frecuentemente se conoce como una medida de purezax, donde un valor pequeño indica que un nodo contiene predominantemente observaciones de una sola clase.

Cuando se aplican modelos de clasificación, a menudo se usa una matriz de confusión para evaluar ciertas medidas de desempeño. Una matriz de confusión (figura 11.7) es una matriz que compara niveles categóricos reales (o eventos) con los niveles categóricos predichos. Cuando se predice el nivel correcto, se dice que se ha poducido un **verdadero positivo** (tp). Sin embargo, si se predice un nivel o evento que no sucedió, esto se denomina **falso positivo** (fn). Por otro lado, cuando no se predijo un nivel o evento y sucede, se denomina **falso negativo** (fn) y si se predijo es un **verdadero negativo** (tn).

A partir de éstos, se pueden extraer diferentes niveles de rendimiento para clasificadores binarios como: - Exactitud ($\frac{tp+tn}{p+n}$). - Precisión ($\frac{tp}{tp+fp}$). - Sensibilidad ($\frac{tp}{p}$). - Especificidad ($\frac{tn}{n}$). - **AUC**: Es el área bajo la curva (*Area Under the Curve*). Un buen clasificador binario tendrá una alta precisión y sensibilidad, por lo que se trata de maximizarlo. Esto significa que el clasificador funciona bien cuando predice que un evento ocurrirá y no ocurrirá, lo que minimiza los falsos positivos y los falsos negativos. Para capturar este equilibrio, a menudo se usa una **curva ROC** (*Receiver Operating Characteristic*) como la de la Fig. 11.8. La curva ROC traza la tasa de falsos positivos a lo largo del eje x y la tasa de verdaderos positivos a lo largo del eje y. Una línea que es diagonal desde la esquina inferior izquierda hasta la esquina superior derecha representa una suposición aleatoria. Cuanto más alta esté la ROC en la esquina superior izquierda, mayor será el área AUC bajo esta curva.

Figura 11.7: Matriz de confusión

Figura 11.8: Curva ROC

RESUMEN

Se ha dado una visión de las principales transformaciones que se realizan en la fase de preprocesamiento del un proyecto de modelado predictivo, la selección de variables (*feature selection*), las transformaciones de la escala o distribución de la variable objetivo, la transformación de variables (*feature engineering*), la reducción de dimensionalidad y otro tipo de tratamientos. La creación de variables predictoras a partir de los datos en bruto tiene una componente creativa, que requiere de herramientas adecuadas y de experiencia para encontrar las mejores representaciones, apoyándose en lo posible en el conocimiento que se tenga de los datos. Se ha utilizado fundamentalmente el paquete `caret` y `rsample` para realizar las tareas de procesamiento sobre dos conjuntos de datos, el de la tienda de "Beauty eSheep" y el de `Madrid_Sale`.

Parte III

Fundamentos de estadística

Capítulo 12

Probabilidad

Mª Leticia Meseguer Santamaría y *Manuel Vargas Vargas*

12.1. Introducción a la probabilidad

La incertidumbre es inevitable en muchos campos científicos, producto de la imposibilidad de predeterminar el resultado de un fenómeno repetido bajo idénticas condiciones, el desconocimiento de todas las causas que pueden influir en él, o una información limitada sobre los condicionantes que rigen su comportamiento. De hecho, gran parte del *"avance científico"* consiste en reducir o controlar el nivel de incertidumbre, bien mejorando el proceso de obtención e interpretación de datos, o estableciendo *"modelos"* que *"expliquen"* los resultados.

Producto de la incertidumbre, las decisiones que se toman (o la validez de los resultados que se obtienen) conllevan un *"riesgo"*, que puede concretarse en enunciados equivocados, modelos con escaso poder predictivo o decisiones con resultados no deseados. Sin embargo, no se prescinde de tomar decisiones en ambientes de incertidumbre, sino que se intenta evaluar y minimizar los riesgos asociados.

Así pues, resulta importante poder *"medir"* la incertidumbre, es decir, cuantificar su magnitud y establecer *"reglas de medida"* que permitan su tratamiento, la estimación de riesgos, y ayuden a la toma de decisiones. La **teoría de la probabilidad** se puede entender como un ente que proporciona reglas de comportamiento que ayudan en los objetivos anteriores, siendo el campo de aplicación tan amplio que puede cubrir cualquier rama de las ciencias sociales, técnicas y naturales.

El concepto de probabilidad apareció en la antiguedad, asociado a los juegos de azar, y se ha ido refinando y formalizando a lo largo de la historia. Sin embargo, la mayoría de las definiciones tradicionales presentan limitaciones que impiden su uso riguroso en cualquier situación. Aún así, siguen estando en el subconsciente colectivo, de forma que se entienden expresiones como *"es muy probable que llueva mañana"*, *"es improbable que me toque la lotería de Navidad"*, o

"es probable que se obtenga en breve una vacuna contra cierta enfermedad" cuando responden a conceptualizaciones diferentes y, en muchos casos, vagas e imprecisas.

Aunque sigue habiendo debates filosóficos y epistemológicos sobre el concepto de probabilidad (véase, por ejemplo, Hajek and Hitchcock (2016)), su uso generalizado en muchos campos científicos está relacionado más con el desarrollo de su carácter de "medida" de la incertidumbre, con un tratamiento matemático que permite su aplicación práctica (véase, por ejemplo, Ross (2012), Morin (2016) o Balakrishnan et al. (2019)). Es este enfoque el que se desarrolla sucintamente en este capítulo.

12.2. Probabilidad: elementos básicos, definición y teoremas

El desarrollo del concepto de probabilidad, entendido como medida de la incertidumbre sobre la ocurrencia de un evento, precisa de algunos requisitos previos que permitan una aplicación operativa.

En primer lugar, es necesario definir en qué situaciones se puede aplicar. Se entenderá por **experimento** cualquier acción u observación de la realidad que pueda repetirse varias veces en idénticas condiciones, dando lugar a resultados identificables y conocidos antes de ser realizado. Cuando, dadas las condiciones, se conoce qué resultado se producirá, se dice que el experimento es **determinista**; en caso contrario, si dadas las condiciones, no se puede saber cuál ocurrirá, el experimento se denomina **aleatorio** o **estocástico**. Así pues, sólo se puede hablar de *"probabilidad"* sobre experimentos aleatorios.

Ahora, dado un experimento aleatorio (E) y el conjunto de posibles resultados (Ω), se define una **probabilidad** como una medida del grado de creencia en la ocurrencia de cada posible resultado, denominado genéricamente **suceso** $S \in \Omega$. Como se ve, la definición es muy amplia, por lo que precisa de algún requisito para evitar que una asignación concreta de grados de creencia produzca inconsistencias. Dicho requisito supone el cumplimiento de una estructura (matemática) concreta, que se adopta de forma axiomática. La más conocida, debida a Andrei Kolmogorov se puede formalizar como:

- **Axiomática de Kolmogorov**: Se considera un experimento aleatorio E, el conjunto de posibles sucesos Ω, y una función real P que asigna a cada suceso un número real. Se dice que P es una medida de probabilidad si cumple:
  - 1. $P(S) \geq 0, \forall S \in \Omega$.
  - 2. $P(\Omega) = 1$
  - 3. Dada una sucesión numerable de sucesos $\{S_i\}$ disjuntos dos a dos $S_i \cap S_j = \emptyset \; \forall i, j$, la probabilidad del suceso unión es la suma de probabilidades, $P\left(\underset{i}{\cup} S_i\right) = \sum\limits_i P(S_i)$.

Así, una probabilidad es una medida que cumple esta axiomática, asignando a suceso un valor real (entre 0 y 1) que expresa el grado de creencia en la ocurrencia de dicho suceso, entendiendo que 0 indica que se cree que no ocurre nunca y 1, que ocurre seguramente (ver, por ejemplo de Finetti (2017) para su fundamentación).

Algunas consecuencias que se derivan de la axiomática de Kolmogorov de forma inmediata son: - $P(\emptyset) = 0$ - Denominando $\bar{S}$ al suceso complementario, $P(\bar{S}) = 1 - P(S)$ - Dados dos sucesos cualesquiera, $P(S_1 \cup S_2) = P(S_1) + P(S_2) - P(S_1 \cap S_2)$

Sin embargo, esta definición es formal, en el sentido de que indica qué requisitos debe cumplir para evitar inconsistencias, pero no determina qué valor concreto de probabilidad asignar a cada suceso. Históricamente, se han propuesto varias concepciones para resolver este problema:

- Concepción **clásica** (o de Laplace): Dado un experimento aleatorio E con n posibles resultados elementales mutuamente excluyentes e igualmente verosímiles, la probabilidad de un suceso S_i es:

$$P(S_i) = \frac{\text{casos favorables en los que ocurre } S_i}{\text{casos posibles}} = \frac{n_i}{n} = f_i \tag{12.1}$$

  Por *"igualmente verosímiles"* se entiende que "no hay razón para afirmar que uno suceda más veces que otro", conocido como *"principio de razón insuficiente"*. Es fácilmente comprobable que esta regla cumple la axiomática de Kolmogorov, e interpreta la probabilidad como la *"frecuencia"* de ocurrencia de cada suceso. A pesar de sus limitaciones (utiliza la equiprobabilidad de los sucesos elementales para definir las probabilidades y asume un conjunto finito de ellos), su fácil comprensión y utilidad en casos sencillos hace que esta regla sea muy utilizada (e incluso confundida con una "definición" de probabilidad).

- Concepción **frecuentista**: Se consideran n repeticiones de un experimento aleatorio, manteniendo idénticas condiciones. Sea n_i el número de veces que se presenta el suceso S_i, entonces se le asigna la probabilidad:

$$P(S_i) = \lim_{n \to \infty} \frac{n_i}{n} \tag{12.2}$$

  Esta concepción, extiende la versión clásica, identificando la probabilidad con la frecuencia relativa de cada resultado cuando el experimento se repite un gran número de veces. Su uso en la práctica se conoce como **la concepción frecuentista de la probabilidad**.

El siguiente paso es formalizar cómo influye la ocurrencia de un suceso sobre la probabilidad de que ocurran otros. Así, dado un suceso A con $P(A) > 0$, la probabilidad de que ocurra otro, B, condicionado a que ha ocurrido A, $P(B/A)$, se calcula como:

$$P(B/A) = \frac{P(B \cap A)}{P(A)} \tag{12.3}$$

es decir, la probabilidad de que ocurran simultáneamente ambos dividida entre $P(A)$ para que $P(\Omega/A) = 1$.

Esta nueva medida se denomina **probabilidad condicionada**[1] y permite obtener resultados fundamentales para el cálculo de probabilidades:

- **Independencia de sucesos**: Dos sucesos, A y B, se dicen independientes si $P(B/A) = P(A) \Rightarrow P(B \cap A) = P(B)P(A)$.
- **Teorema de la probabilidad total**: Dado un conjunto de sucesos $\{A_i\}$ disjuntos y cuya unión sea Ω (denominado **partición** de Ω), la probabilidad de cualquier suceso es:

$$P(B) = \sum_i P(B/A_i)P(A_i) \tag{12.4}$$

  Este teorema permite determinar la probabilidad de un suceso B, que puede tener varias causas A_i, mediante la suma de las probabilidades de que aparezca B condicionada a cada una de las causas ponderada por la probabilidad de cada causa.

- **Teorema de Bayes**: Dada una partición de Ω, y un suceso B con $P(B) > 0$, la probabilidad condicionada de cada elemento de la partición a que ha ocurrido B es:

$$P(A_i/B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(B/A_i)P(A_i)}{\sum_j P(B/A_j)P(A_j)} \tag{12.5}$$

  Este teorema, aplicación directa de la definición de probabilidad condicionada y del teorema de la probabilidad total, es un resultado tan importante que su uso ha dado nombre a una rama entera de la estadística, la conocida como **estadística bayesiana**.[2]

12.3. Variable aleatoria y su distribución: tipos de variables aleatorias

Una limitación operativa de la probabilidad tal como se ha utilizado hasta ahora es que hace referencia a sucesos y operaciones entre conjuntos, lo que dificulta su tratamiento. Sin embargo, en muchos casos los sucesos están caracterizados por valores numéricos, por lo que podrían ser utilizados en sustitución de los primeros para facilitar los cálculos. A esta idea corresponde la noción de **variable aleatoria** (v.a.) que es una función que asigna un valor numérico a cada suceso de un experimento aleatorio. Para trabajar con probabilidades sobre números, a cada uno se le asigna la probabilidad de los sucesos que están caracterizados por dicho valor. [3]

Dada una v.a. X, su **función de distribución** asigna a cada número real x la probabilidad de que la variable tome un valor menor o igual que x, $F_X(x) = P(X \leq x)$.

[1] Esta definición cumple la axiomática de Kolmogorov y es la forma de introducir "información" en la determinación de probabilidades. Su versión para distribuciones de probabilidad es muy utilizado en inferencia estadística.

[2] $P(A_i)$ reciben el nombre de **probabilidades a priori**, $P(B/A_i)$ el de **verosimilitudes**, y $P(A_i/B)$, **probabilidades a posteriori**. El teorema establece cómo se modifican las probabilidades cuando se introduce información en forma de verosimilitudes, siendo muy utilizada su versión para distribuciones de probabilidad.

[3] Matemáticamente, una variable aleatoria es una función $X : \Omega \longrightarrow \mathbb{R}$ que, para cada valor real, cumple $X^{-1}(x) \in \Omega$, de forma que se pueda asignar $P(x) = P\left(X^{-1}(x)\right)$.

Una variable de dice **discreta** si sólo puede tomar un conjunto finito (o infinito numerable) valores con probabilidad positiva. A ese conjunto de valores y sus probabilidades $\{x_i\}$, $P(X = x_i)$ se le denomina **función de cuantía**.

Una variable se denomina **continua**, si su función de distribución es continua, existe la primera derivada y es continua. Como consecuencia, la probabilidad en un valor concreto siempre será cero, $P(X = x_i) = 0$, por lo que sólo habrá probabilidades positivas sobre intervalos. Se denomina **función de densidad** a la derivada de la función de distribución $f(x) = F\text{t}(x) = \frac{dF(x)}{dx}$.[4]

Ya que una v.a. X está caracterizada por su distribución (a través de la función de distribución o por la de cuantía-densidad), se han desarrollado **modelos de distribución de probabilidad** que permiten modelizar el comportamiento aleatorio de las variables y calcular probabilidades de forma fácil.

12.4. Modelos de distribución de probabilidad

Se van a presentar los modelos más habituales en la práctica (para una visión completa, se puede consultar Johnson et al. (2008)). En este punto se presentan algunos modelos de distribuciones de probabilidad según la v.a. sea discreta o continua.

12.4.1. Modelos discretos

Los modelos de distribución discretos más habituales son el Binomial, el Binomial Negativo y el de Poisson. Los dos primeros se asientan sobre el **fenómeno de Bernoulli**, que, de manera general, consiste en n repeticiones independientes de un experimento dicotómico, es decir, que tiene sólo dos posibles resultados: uno identificado con el *"éxito"* del experimento, cuya probabilidad se denota por p, y el otro, con el *"fracaso"*, con probabilidad $q = (1 - p)$.

- **Distribución Binomial B(n;p)**

La distribución binomial es una distribución de probabilidad discreta que mide el número de éxitos en una secuencia de n *ensayos* independientes de Bernoulli con una probabilidad fija de éxito p. Puede tomar los valores $x = 0, 1, ..., n$ y su función de cuantía es:

$$P(X = x) = \binom{n}{x} p^x q^{n-x} \equiv \frac{n!}{x!(n-x)!} p^x q^{n-x} \tag{12.6}$$

Su media es $E(X) = \mu = np$ y su varianza, $Var(X) = \sigma^2 = npq$

La representación gráfica de las funciones de cuantía y distribución se muestra en la fig. 12.1

En el **caso particular** de que n=1, B(1;p), se denomina **Distribución Bernouilli, B(p)**.

[4] La función de densidad es el equivalente continuo de la función de cuantía, indicando, de forma intuitiva, dónde se "concentra" la probabilidad de observar valores de X. De hecho, no es raro que se utilize el término general de "densidad" independientemente del tipo de variable que sea.

Figura 12.1: Función de cuantía y de distribución para una variable B(10,0.5)

- **Distribución Binomial Negativa o de Pascal BN(r,p)**

La distribución Binomial Negativa es una serie de ensayos de Bernoulli independientes, con probabilidad constante de éxito p, donde la v.a. X denota el número de ensayos fracasados (x) hasta que se produce un número determinado de éxitos (r). Puede tomar los valores $x = 0, 1, ...$ y su función de cuantía es:

$$P(X = x) = \binom{x + r - 1}{x} q^x p^r \equiv \frac{(x + r - 1)!}{(r - 1)!x!} q^x p^r \tag{12.7}$$

con media: $E(X) = r\frac{q}{p}$ y varianza $Var(X) = r\frac{q}{p^2}$.

La representación gráfica de las funciones de cuantía y distribución se muestra en la fig. 12.2

En el **caso particular** de que r=1, BN(1;p), se está ante una **Distribución Geométrica G(p)**.

- **Distribución de Poisson** $P(\lambda)$

Se denominan **procesos de Poisson** a aquéllos en los que la ocurrencia de un suceso se encuentra distribuida a lo largo de un tiempo (o espacio) dado, cumpliendo que el proceso es estable, con una media de ocurrencias λ por unidad de tiempo, que se presentan de forma aleatoria e independiente. La variable que mide el número x de ocurrencias puede tomar los valores $x = 0, 1, 2, ...$ y se dice que sigue una distribución de Poisson, con función de cuantía:

$$P(X = x) = \frac{e^{\lambda}\lambda^x}{x!} \tag{12.8}$$

Su media es $E(X) = \lambda$ y su varianza, $Var(X) = \lambda$.

La representación gráfica de las funciones de cuantía y distribución se muestra en la fig. 12.3

Figura 12.2: Función de cuantía y de distribución para una variable BN(3,0.35)

Figura 12.3: Función de cuantía y de distribución para una variable P(2.5)

12.4.2. Modelos continuos

Los modelos de distribución para variables continuas más habituales son el Normal, Gamma, Chi-cuadrado, t-student y F-Snedecor.

- **Distribución Normal** $N(\mu, \sigma)$

La distribución Normal, de Gauss o gaussiana, tiene una gran importancia debido a que un gran número de fenómenos aleatorios se pueden modelizar a partir de esta distribución (véase, más adelante, el epígrafe 12.5.

Una v.a. se dice que sigue una distribución normal de parámetros μ y σ si puede tomar cualquier valor real y su función de densidad es de la forma:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{12.9}$$

Su media es $E[X] = \mu$ y su varianza, $Vas(X) = \sigma^2$. Por ello, se suele decir que la normal está caracterizada por su media y su desviación típica.

La gráfica de la función de densidad tiene forma de campana (conocida como **campana de Gauss**) y es simétrica respecto de la media, y con colas mayores conforme aumenta la desviación típica, como muestra la fig. 12.4, donde $\mu = 0$ y $\sigma = 1, 1{,}5,$ y 2

Figura 12.4: Función de densidad y de distribución para variable Normales

Una característica importante de la distribución normal es la **propiedad aditiva o reproductiva**, que indica que la combinación lineal de distribuciones normales sigue siendo una distribución normal. Si se consideran n variables aleatorias independientes con distribuciones $N(\mu_i, \sigma_i)$ cualquier combinación lineal cumple:

$$\beta_0 + \beta_1 X_1 + ... + \beta_n X_n = \beta_0 + \sum_{i=1}^{n} \beta_i X_i \sim N\left(\beta_0 + \sum_{i=1}^{n} \beta_i \mu_i, \sqrt{\sum_{i=1}^{n} \beta_i^2 \sigma_i^2}\right) \tag{12.10}$$

En particular, si $X \sim N(\mu, \sigma)$, la variable $Z = \frac{X-\mu}{\sigma} \sim N(0,1)$, conocida como **normal estándar** o **normal tipificada**.[5]

- **Distribución Gamma** $\Gamma(\alpha, \beta)$

La distribución Gamma es útil cuando se medir el tiempo de espera (o la vida útil) hasta que ocurre un número determinado de eventos (α), suponiendo que β (a veces también representado por θ o parámetro de escala) es el tiempo medio entre ocurrencias de un suceso.

Esta distribución toma valores positivos y su función de densidad viene dada por la expresión:

$$f(x) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta} \tag{12.11}$$

donde $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx = (\alpha - 1)!$ si α es un número natural.

Su media es $E(X) = \alpha\beta$ y su varianza, $Var(X) = \alpha\beta^2$. El parámetro α es conocido como **parámetro de forma**; si $\alpha \leq 1$ la función de densidad tiene forma de"distribución gamma tiene forma de "L"; si $\alpha > 1$, la distribución es campaniforme, con asimetría positiva y, conforme va aumentando, el centro de la distribución se desplaza hacia la derecha. β se conoce como "parámetro de escala" y determina el alcance de la asimetría positiva. La fig. 12.5 muestra la representación gráfica de las funciones de densidad y distribución para $\beta = 2$ y $\alpha = 1$ (azul), $\alpha = 2$ (morado), $\alpha = 5$ (rojo) y $\alpha = 10$ (verde):

Figura 12.5: Función de densidad y de distribución para variables Gamma

En el **caso particular** de que $\alpha = 1$, se denomina **distribución exponencial** de parámetro β.

- **Distribución χ^2_n de Pearson**

[5] También es posible interpretar que toda distribución normal es una transformada de la distribución Z, ya que $X = \mu + \sigma Z$.

Sean $X_1, X_2, ..., X_n$ v.a.i. todas distribuidas según una N(0,1). La suma de sus cuadrados sigue una distribución $\Gamma(\frac{n}{2}, 2)$, que se denomina **distribución Chi-cuadrado**: [6]

$$\sum_{i=1}^{n} X_i^2 \sim \chi_n^2 \equiv \Gamma(\frac{n}{2}, 2) \tag{12.12}$$

Al parámetro n, se le llama **grados de libertad**; su media es $E(X) = n$ y su varianza, $Var(X) = 2n$ y la forma funcional de su densidad y distribución son casos particulares de la Gamma. De hecho, la fig. 12.5 corresponde a distribuciones χ_n^2 con $n = 2, 4, 10,$ y 20.

- **Distribución** $t - Student$

La distribución $t - Student$ es otra distribución de probabilidad que surge en el muestreo de poblaciones normales (véase el epígrafe 13.6), en este caso asociada al uso de medias. Se dice que una v.a. X sigue una distribución t-student, con n-grados de libertad, si es el cociente entre una distribución normal estándar y la raiz de una χ_n^2 dividida entre sus grados de libertad $X \sim \frac{N(0,1)}{\sqrt{\chi_n^2/n}}$

Su función de densidad viene dada por:

$$f(x) = \frac{\Gamma((n+1)/2)}{\sqrt{n\pi}\Gamma(n/2)}(1 + x^2/n)^{-(n+1)/2} \tag{12.13}$$

con media $E(X) = 0$ y varianza, $Var(X) = \frac{n}{n-2}$, siendo n>2. Su densidad tiene forma acampanada, simétrica respecto a cero y parecida a la de la Normal, pero con mayor probabilidad en las colas. En la fig. 12.6 se muestran las funciones de densidad y distribución para tres $t - Student$, con 3 (verde), 10 (rojo) y 100 (azul) grados de libertad

- **Distribución** F **de Snedecor**

Este modelo también está asociado al muestreo sobre poblaciones normales, en este caso, a la comparación de varianzas. Se define una distribución F de Snedecor con (n, m) grados de libertad como el cociente de dos distribuciones χ^2 independientes divididas entre sus grados de libertad, $F_{n,m} \sim \frac{\chi_n^2/n}{\chi_m^2/m}$

La función de densidad $F_{n,m}$ viene dada por:

$$f(x) = \frac{\Gamma(\frac{n+m}{2})}{\Gamma(\frac{n}{2})\Gamma(\frac{m}{2})}\left(\frac{n}{m}\right)^{\frac{n}{2}} \frac{x^{\frac{n-2}{2}}}{(1 + \frac{nx}{m})^{\frac{n+m}{2}}} \tag{12.14}$$

con media $E(X) = \frac{m}{m-2}$, siendo m>2 y varianza $Var(X) = \frac{2m^2(n+m-2)}{n(m-2)^2(m-4)}$, cuando m>4

La gráfica de la función de densidad es parecida a la de la χ_2. Así, sólo está definida para el semieje positivo y su apariencia variará según los grados de libertad. La fig. 12.7 muestra las funciones de densidad y distribución para varias distribuciones F-Snedecor:

[6] Como se verá en el capítulo siguientes, epígrafe 13.6, esta distribución está aparece en el muestreo de poblaciones normales, en concreto al trabajar con varianzas, lo que justifica su interés.

Figura 12.6: Función de densidad y de distribución para variables t-Student

Figura 12.7: Función de densidad y de distribución para variables F-Snedecor

12.5. Teorema central del límite

A veces es difícil encontrar la distribución muestral de algunos estadísticos o estimadores, o, incluso, es imposible determinar la distribución de la variable de interés; entonces es útil aplicar algunos teoremas de convergencia, en especial el Teorema Central del Límite (TCL).

El TCL permite, bajo ciertas condiciones, usar la distiribución normal para estudiar otras distribuciones más generales.

Teorema Central del Límite: Si $X_1, ..., X_n$ son v.a.i. e idénticamente distribuidas con media μ y desviación típica σ, entonces $\sum_{i=1}^{n} X_i$ tiene asintóticamente una distribución normal de media $n\mu$ y desviación típica $\sqrt{n}\sigma$. Y, por las propiedades de la normal, tenemos que:

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma} \underset{n\to \text{ínf}}{\longrightarrow} N(0,1) \tag{12.15}$$

Señala que la distribución de la suma de n variables aleatorias tiende a una distribución normal, cuando n es muy grande. Es decir, aunque cada uno de los efectos sea raro o difícil de estudiar, si lo que queremos estudiar es la suma de los mismos sabemos que, bajo ciertas condiciones, esta se comportará como una distribución normal. Así, se explica el hecho constatado de que muchas distribuciones de variables observadas en la naturaleza o en experimentos físicos sean aproximadamente normales, por ejemplo, las medidas del cuerpo humano, altura, peso, longigtud de los dedos, etc.

12.6. Distribuciones de probabilidad en R

En **R** están definidas las distribuciones de probabilidad más importantes, de tal forma que se aplica a cada nombre del modelo un determinado prefijo para calcular una función específica, **"d"** para la función de cuantía o densidad, **"p"** para la función de distribución, **"q"** para los cuantiles (o percentiles) y **"r"** para generar muestras pseudo-aleatorias.

Se va a utilizar el paquete **Rlab**, por lo que lo primero es instalar este paquete y cargar la librería.

```
library(Rlab)
```

En la tabla siguiente se exponen los modelos de distribución vistos, indicando el tipo, con su notación y la función utilizada en R para su cálculo:

Tabla : Funciones de distribución en **R**

Distribución	Tipo de modelo	Notación	Función en **R**
Binomial	discreto	B(n,p)	binom
Bernoulli	discreto	B(p)	bern
Binomial negativa	discreto	BN(r;p)	nbinom

Distribución	Tipo de modelo	Notación	Función en **R**
Geométrica	discreto	G(p)	geom
Poisson	discreto	P(λ)	pois
Normal	continuo	$N(\mu, \sigma)$	norm
Gamma	continuo	$\Gamma(\alpha, \beta)$	gamma
Exponencial	continuo	$Exp(\beta)$	exp
Chi-cuadrado	continuo	χ^2_n grados de libertad	chisq
t-student	continuo	t_n grados de libertad	t
F-Snedecor	continuo	$F_{n,m}$ grados de libertad	f

A continuación se realizarán dos ejemplos con **R**, uno para modelos discretos y otro para la normal. La adaptación a cualquier otro modelo de probabilidad consiste, básicamente, en sustituir las funciones de **R**.

Ejemplo con distribuciones discretas:

Un algoritmo de identificación de imágenes tiene acreditada una tasa de error del 20 % en caso de personas y del 5 % para el resto de imágenes, suponiendo el primer tipo el 25 % del total de imágenes.

a) Si se analizan 10 imágenes de personas, calcular la probabilidad de que identifique correctamente siete.

Denominando X al número de imágenes de personas correctamente clasificadas, se tiene que $X \sim B(10, 0{,}8)$. Se pide $P(X = 7) =$

```
dbinom(7, size = 10, prob = 0.8)
#> [1] 0.2013266
```

b) Para el resto de imágenes, calcular la probabilidad de identificar correctamente como mucho 50 hasta que se produce el segundo error.

Denominando Y al número de imágenes correctamente identificadas hasta el segundo error, se tiene que $Y \sim BN(2, 0{,}05)$. Se pide $P(Y \leq 50)$

```
pnbinom(50, size = 2, prob = 0.05)
#> [1] 0.7405031
```

c) Históricamente, el número medio diario de imágenes incorrectamente clasificadas es de 7. Calcular la probabilidad de que un día seleccionado al azar clasifique incorrectamente entre 6 y 9 imágenes.

Denominando T al número de imágenes incorrectamente identificadas en un día, se tiene que $T \sim P(\lambda = 7)$. Se pide $P(6 \leq T \leq 9) = P(T \leq 9) - P(T \leq 5)$

```
ppois(9, 7, lower.tail = TRUE) - ppois(5, 7, lower.tail = TRUE)
#> [1] 0.5297877
```

d) Calcular la probabilidad de que en un lote de 20 imágenes del mismo tipo sean todas correctamente clasificadas.

Como no se especifica el tipo de imágenes se identifica, hay que calcular dicha probabilidad condicionada a cada grupo y utilizar el teorema de la probabilidad total: $P(acierto) = P(acierto/personas) * P(personas) + P(acierto/otras) * P(otras)$

```
acierto_personas <- dbinom(0, 20, 0.2)
acierto_otras <- dbinom(0, 20, 0.05)
acierto_total <- acierto_personas * 0.25 + acierto_otras * 0.75
acierto_total
#> [1] 0.2717467
```

e) Si se han clasificado correctamente las 20 imágenes del lote, calcular la probabilidad de que correspondan a imágenes de personas.

Se pide la probabilidad de que correspondan a personas condicionado a que han sido correctamente clasificadas, por lo que hay que utilizar el teorema de Bayes: $P(personas/acierto) = \frac{P(acierto/personas)*P(personas)}{P(acierto)}$

```
acierto_personas * 0.25 / acierto_total
#> [1] 0.01060658
```

Ejemplo con una distribución normal:

Las calificaciones (de 0 a 10) en un curso de estadística siguen una de distribución normal N(6, 1.25), $X \sim N(6, 1{,}25)$

a) Calcular la probabilidad de que una persona obtenga una calificación inferior a 4.

```
pnorm(4, mean = 6, sd = 1.25)
#> [1] 0.05479929
```

b) Número probable de personas que obtendrán sobresaliente (9 o más) en un grupo de 60 personas.

```
p <- pnorm(9, 5, 1.25, lower.tail = FALSE)
p
#> [1] 0.0006871379
# En un grupo de 60 personas, redondeando a un número entero
round(60 * p)
#> [1] 0
```

c) Calcular la nota mínima para estar en el 30 % de personas con mejores calificaciones.

```
qnorm(0.7, 6, 1.25)
#> [1] 6.655501
```

d) En un curso de informática las calificaciones siguen una de distribución normal N(5, 1.75), independientes de las de estadística. Calcular la probabilidad de que una persona matriculada en ambos cursos saque mayor calificación en estadística.

Llamando $X = C_e - C_i$ a la diferencia entre las calificaciones en estadística (C_e) y en informatica (C_i), ambas normales e independientes, la distribución de X será $X \sim N\left(6-5, \sqrt{1{,}25^2 + 1{,}75^2}\right)$. Se pide $P(X > 0)$, que se representa en la fig. 12.8

```
pnorm(0, mean = 1, sd = sqrt(1.25^2 + 1.75^2), lower.tail = FALSE)
#> [1] 0.6790309
```

Figura 12.8: P(X>0) representada como el área bajo la densidad de la distribución normal

RESUMEN
La teoría de la probabilidad proporciona reglas de comportamiento que permiten la ordenación y toma de decisiones en situaciones donde prevalecen condiciones de incertidumbre.
Los modelos de distribuciones de probabilidad permiten el cálculo en, prácticamente, cualquier estudio de las ciencias sociales, técnicas y naturales.

Capítulo 13

Inferencia estadística

Mª Leticia Meseguer Santamaría y *Manuel Vargas Vargas*

13.1. Introducción a la Inferencia Estadística

Cuando se estudian fenómenos mediante variables, el objetivo estadístico básico es determinar cuáles son las distribuciones probabilísticas que rigen dichas variables o algunas características determinadas por ellas. Es este comportamiento aleatorio el que permite hacer predicciones con unos márgenes de error conocidos, analizar y cuantificar la relación entre variables, evaluar si hipótesis o modelos teóricos son congruentes con los datos disponibles, etc. Así, en la práctica, cuando se estudia una variable, X, lo habitual es que se desconozca su distribución, llamada distribución poblacional, $F(x)$, pero que se disponga de un conjunto de realizaciones $(x_1, ..., x_n)$, también llamado muestra, valores concretos de dicha variable a partir de los cuales "aproximar" la distribución desconocida.

La inferencia estadística proporciona las herramientas y técnicas que permiten, a partir de la información muestral, extrapolar resultados a la distribución poblacional con márgenes de error conocidos. Un primer objetivo (más detallado en el Capítulo siguiente, 14) es analizar qué condiciones debe cumplir la muestra para que su información sea válida y extrapolable a toda la población (es la conocida como **teoría de muestreo**). Un segundo objetivo es establecer los mecanismos que permitan dicha extrapolación manteniendo controlados los errores de muestreo.

Es habitual que se conozca (o se asuma) que la distribución poblacional $F(x)$ pertenezca a alguna familia paramétrica, es decir, que se asuma su forma funcional pero que dependa de algunos parámetros (lo más frecuente es que se asuma la normalidad, pero podría ser cualquiera de los modelos paramétricos existentes). Se habla entonces de **inferencia paramétrica**, ya que se usa la información muestral para determinar los "mejores" valores (bajo algún criterio) de los parámetros que rigen la distribución poblacional, existiendo tres planteamientos básicos, estimación puntual (epígrafe 13.3), por intervalo (epígrafe 13.4) y contraste de hipótesis (epígrafe 13.5).

También hay situaciones en las que la forma funcional de la distribución poblacional es desconocida, o se duda de que la familia paramétrica considerada sea adecuada. En estos casos, bajo el nombre genérico de **inferencia no paramétrica**, se plantean contrastes que buscan determinar cuándo es posible asumir un modelo concreto de distribución, entre los que destacan, por su frecuente uso, los contrastes de normalidad (epígrafe 13.8). Otra alternativa que permite aproximar características poblacionales sin asumir ninguna distribución poblacional concreta es el **remuestreo**, fundamentalmente el denominado *"bootstrap"*, (tema tratado en el Capítulo 14).

13.2. Muestreo aleatorio simple

Al estudiar una variable poblacional, X, de la que se desconoce su distribución, llamada **distribución poblacional,** $F(x)$, se utiliza la información suministrada por una **muestra** obtenida por algún método de muestreo probabilístico que garantice que sea representativa de la variable poblacional.

En la mayoría de los casos y técnicas estadísticas se asume que la muestra está obtenida mediante el método básico de muestreo, conocido como **muestreo aleatorio simple**, consistente en seleccionar totalmente al azar a los individuos de la muestra, por lo que todos tienen la misma probabilidad de formar parte de ella. Si cada individuo sólo puede aparecer una vez en la muestra, se habla de **muestra sin reemplazamiento**, mientras que se denomina **muestra con reemplazamiento** en caso contrario. De esta forma, dada una distribución poblacional $F(x)$,una muestra aleatoria simple (m.a.s.) es una realización de un conjunto de n variables aleatorias independientes e idénticamente distribuidas $X = (X_1, ..., X_n)$, cuya **distribución conjunta** es de la forma:

$$F(X_1, ..., X_n) = F_{X_1}(x_1)...F_{X_n}(x_n) = F(x_1)...F(x_n) \tag{13.1}$$

Una herramienta básica para la inferencia es la **distribución empírica de la muestra**, definida como:

$$\hat{\mathbb{F}}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{(-\infty,x]}(X_i) \tag{13.2}$$

donde por $\mathbb{I}_{(-\infty,x]}(X_i)$ se indica la función indicadora, que toma el valor 1 si $X_1 \leq x$ y 0 en caso contrario.[1]

La gran ventaja del muestreo aleatorio simple consiste en que, dada una m.a.s. $(X_1, ..., X_n)$:

$$\lim_{n\to\infty} E\left[\left(\hat{\mathbb{F}}_n(x) - F(x)\right)^2\right] = 0 \tag{13.3}$$

conocido como *teorema de Glivenko-Cantelli*. Este resultado es fundamental en inferencia, pues garantiza que el muestreo aleatorio simple produce muestras representativas de la población, ya que, a medida que aumenta el tamaño muestral, la distribución empírica de la muestra se

[1] Es decir, la distribución empírica de la muestra indica, para cada valor x, la proporción de elementos de la muestra que toman un valor menor o igual que él.

aproxima a la distribución poblacional (véase fig. 13.1). [2] Así, cualquier característica de una distribución poblacional puede ser aproximada por su equivalente en la distribución empírica.

Figura 13.1: Distribución empírica para muestras de diferente tamaño de una distribución Normal

Es muy frecuente que, a efectos de inferencia, no se estudie el comportamiento aleatorio de toda la muestra (su distribución conjunta) sino que interese el comportamiento de una función de la muestra que no dependa de ningún valor desconocido, $T(X) = T(X_1, ..., X_n)$, llamada genéricamente **estadístico muestral**; dicho comportamiento vendrá determinado por la **distribución en el muestreo** del estadístico $T(X)$. El hecho de utilizar una m.a.s. permite establecer resultados de interés sobre los estadísticos o, en algunos casos, incluso obtener la distribución en el muestreo exacta de los estadísticos más usuales (epígrafe 13.6).

Así, dada una variable poblacional X con varianza finita, y una m.a.s. se define la **media muestral** como

$$\bar{X} = \frac{X_1, +... + X_n}{n} \tag{13.4}$$

El hecho de utilizar una m.a.s. garantiza que:

$$E[\bar{X}] = E[X] \; ; \; Var(\bar{X}) = \frac{Var(X)}{n} \tag{13.5}$$

Este resultado es muy útil, ya que indica que la variabilidad de la media muestral es más pequeña que la variabilidad de la variable poblacional, siendo inversamente proporcional al tamaño muestral.

Otro estadístico muy utilizado es la **varianza muestral**, que se define como:

$$S^2 = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2}{n} \tag{13.6}$$

[2] Un tema que se abordará en el Capítulo 14 es la determinación del tamaño muestral necesario para que la aproximación tenga un error menor que uno prefijado.

En este caso, su esperanza es:

$$E[S^2] = \frac{n-1}{n} Var[X] \tag{13.7}$$

que no coincide con la varianza poblacional. Para evitar este hecho, se define la **cuasivarianza muestral**:

$$S_c^2 = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2}{n-1} \tag{13.8}$$

estadístico para el que sí se cumple que $E[S_c^2] = Var[X]$, ya que existe una relación de proporcionalidad entre ambos estadísticos $nS^2 = (n-1)S_c^2$.[3]

13.3. Estimación puntual

Sea una población caracterizada por una distribución poblacional $F_\theta(x)$, de una familia paramétrica pero que se desconoce el valor del parámetro $\theta \in \Theta$, donde Θ es el espacio paramétrico (conjunto de posibles valores de θ). Dada una m.a.s. $X = (X_1, ..., X_n)$, se considera como *estimador* de θ a un estadístico muestral cuyo resultado sea un posible valor del parámetro:

$$\hat{\theta} = T(X) = T(X_1, ..., X_n) \in \Theta() \tag{13.9}$$

Para cada estimador, se puede definir su **error cuadrático medio**:

$$ECM_\theta(\hat{\theta}) = E_\theta\left[\left(\hat{\theta} - \theta\right)^2\right] \tag{13.10}$$

que proporciona un valor medio del error que se comete al "aproximar" el verdadero valor θ por el resultado del estimador $\hat{\theta}$. Así, el criterio de *"mínimos cuadrados"* propone utilizar aquel estimador que minimiza el error cuadrático medio:

$$\hat{\theta}_{MC} = \underset{\hat{\theta}}{min} E_\theta\left[\left(\hat{\theta} - \theta\right)^2\right] \tag{13.11}$$

Desarrollando la expresión del ECM (@(eq:ecm)), éste se puede reexpresar como

$$ECM_\theta(\hat{\theta}) = Var_\theta(\hat{\theta}) + \left(E_\theta(\hat{\theta}) - \theta\right)^2 = Var_\theta(\hat{\theta}) + b_\theta^2(\hat{\theta}) \tag{13.12}$$

donde $b_\theta(\hat{\theta}) = \left(E_\theta(\hat{\theta}) - \theta\right)$ se conoce como **sesgo** del estimador (*bias*, en inglés). Así, el ECM de un estimador depende de su varianza y de su sesgo al cuadrado.

Para determinar, por tanto, el *"mejor"* estimador bajo el criterio de mínimos cuadrados, se podría obtener en dos pasos:

- Utilizar estimadores "insesgados", es decir, de sesgo cero, o sea, $E(\hat{\theta}) = \theta$ (el valor medio del estimador coincide con el parámetro).

[3] Muchos textos, sobre todo anglosajones no hacen esta distinción, sino que denominan directamente *"varianza muestral"* a la cuasivarianza. En R, por ejemplo, las funciones *var* o *sd* proporcionan la cuasivarianza y cuasidesviación típica muestrales respectivamente, matiz que hay que tener siempre presente.

- De entre los estimadores inesgados, utilizar el de varianza mínima, $Var(\hat{\theta}_{MC}) = \underset{\hat{\theta}}{min} Var(\hat{\theta})$

Queda fuera del objetivo de este capítulo plantear la obtención del estimador de mínimos cuadrados para cualquier distribución poblacional y parámetros, que el lector interesado puede encontrar en cualquier texto teórico de inferencia estadística (Casella and Berger (2007) , Blais (2020) , Almudevar (2021)).

Otro planteamiento para encontrar estimadores puntuales se basa en la función de densidad de la muestra, que depende de ésta y del parámetro que caracteriza a la distribución poblacional:

$$f(x_1, ..., x_n; \theta) = f(x_1; \theta)...f(x_n; \theta) = L(\theta; x_1, ..., x_n) \tag{13.13}$$

Considerando el parámetro como fijo, la función se interpreta como la densidad de la muestra. Sin embargo, si se considera que la muestra está dada, también se puede interpretar como una función del parámetro, que mide la **verosimilitud** (*likelihood*, en inglés) de cada valor del parámetro en función de la muestra obtenida. Así, el criterio para determinar el *"mejor"* estimador puede ser utilizar aquél que maximiza la función de verosimilitud; se obtiene entonces el conocido como *estimador máximo verosímil*:

$$\hat{\theta}_{MV} = \underset{\theta}{max} L(\theta; x_1, ..., x_n) \tag{13.14}$$

Para el cálculo del estimador máximo verosímil no se suele utilizar la función de verosimilitud, sino su logaritmo (que alcanza los máximos y mínimos en los mismos puntos), derivando respecto al parámetro e igualando a cero (ecuación de verosimilitud).

Este método suele proporcionar estimadores con buenas propiedades estadísticas y, en muchos casos, suele conducir al mismo resultado que el método de mínimos cuadrados. [4]. En las distribuciones usuales, es relativamente sencillo obtener la ecuación de verosimilitud y resolverla, por lo que se dispone de estimadores máximo verosímiles conocidos. En modelos más elaborados, la resolución de la ecuación de verosimilitud se puede complicar, hasta el extremo de que haya que recurrir a métodos numéricos de aproximación.

Una alternativa computacionalmente más sencilla es la basada en el conocido como **método de los momentos**. El planteamiento básico es expresar el parámetro en función de los momentos poblacionales (esperanza, varianza, etc.) y utilizar como estimador la misma función pero de los momentos muestrales (media muestral, varianza muestral, etc.). En las distribuciones más usuales, los parámetros suelen ser momentos poblacionales o tranformadas simples de éstos, por lo que el método de los momentos es muy sencillo. Como contrapartida, es más difícil evaluar las propiedades estadísticas de estos estimadores, salvo que coincidan con los de mínimos cuadrados o de máxima verosimilitud.

En **R**, la librería `fdistrplus` dispone de la función `fitdist()`, que permite la obtención de los estimadores para las distribuciones usuales por diversos métodos, incluidos el de máxima verosimilitud (mle) y el de los momentos (mme).

[4] En las distribuciones usuales es así, salvo que el estimador de máxima verosimilitud no sea insesgado, como es el caso de estimar la varianza en una distibución normal.

13.4. Estimación por intervalos

Dado que todo estimador es una variable aleatoria, su valor concreto, la *"estimación"* de parámetro $\hat{\theta}$, depende de la muestra. Esta variación muestral ocasiona incertidumbre sobre la estimación. Una forma de incluir esta variabilidad en la estimación puede consistir en sustituir la estimación puntual por un intervalo de valores en el que se tenga un cierto nivel de confianza de que contenga al verdadero valor del parámetro.

El método más extendido para obtener **intervalos de confianza** consiste en utilizar un estimador puntual y utilizar su distribución en el muestreo para construir una región que contenga, con cierta probabilidad $(1-\alpha)$, al verdadero valor θ:

$$IC_{(1-\alpha)} = [LIC, LSC] \text{ tal que } P\left(LIC \leq \theta \leq LUC\right) = (1-\alpha) \tag{13.15}$$

donde los límites inferior (LIC) y superior (LSC) de confianza dependen de la desviación típica del estimador y de constantes asociadas a su distribución y al nivel de confianza $(1-\alpha)$, llamados **valores críticos**. En esta ecuación, tanto el LIC como el LSC son variables aleatorias; cuando se utilizan los datos de una muestra, se convierten en valores reales, por lo que no se puede hablar de *"probabilidad de que el parámetro esté dentro del intervalo"*, sino que se habla de *"confianza en que el intervalo contenga al valor del parámetro"*.

En **R**, la librería `Rlab` permite obtener los valores críticos de las distribuciones usuales a través de los cuantiles, anteponiendo q al nombre de la distribución, recogidos en la tabla XXX-ref. cruzada "Funciones de distribución en R" (por ejemplo usando las funciones `qbinom()`, `qnorm()`, `qt()`, `qf()`, etc.). Igualmente, la librería `DescTools` dispone de funciones para calcular intervalos de confianza en poblaciones normales para la media (`MeanCI()`), la diferencia de medias (`MeanDiffCI()`), la mediana (`MedianCI()`), cualquier cuantil (`QuantileCI()`), o la varianza (`VarCI()`). Por último, en el caso de no conocer la distribución en el muestreo del estimador, se puede recurrir al remuestreo por bootstrap, que se detallará en el capítulo 14, indicando el método *"boot"* en las funciones anteriores.

13.5. Contrastes de hipótesis

Hay situaciones donde no interesa tanto estimar el valor de un parámetro sino decidir si la información muestral es congruente con algún valor concreto del parámetro. En estos casos, se puede establecer como **hipótesis** que el parámetro toma un valor concreto y *contrastar* si es verosímil haber obtenido el resultado muestral dado. Este planteamiento se conoce como **contrastes de significación**.

Así, se establece una hipótesis, históricamente conocida como **hipótesis nula**, que determina un valor del parámetro:

$$H_0 \equiv \theta = \theta_0 \tag{13.16}$$

Con este valor, la distribución muestral del estimador permite obtener la probabilidad de observar un valor tan *"distante"* como el obtenido en la muestra, conocido como **p-valor**: si es muy pequeño, sería poco probable haber observado ese valor si la hipótesis es cierta, por lo que la

evidencia empírica no es congruente con ella; si no es pequeño, sería probable haber observado el valor bajo la hipótesis, por lo que no hay evidencia empírica *"contra"* dicha hipótesis nula.

Cuando por *"distante"* se entiende alejado del valor medio del estimador, mayor o menor, se habla de **p-valor bilateral** o "a dos colas". Cuando se considera *"distante"* como menor que (o mayor que) la media del estimador, se habla de **p-valor unilateral** (a la izquierda o a la derecha, respectivamente) o "a una cola".

Habitualmente, se considera que un p-valor por debajo de 0.05 ya indica que la evidencia empírica no permite asumir como cierta la hipótesis nula, expresándose como que el valor del parámetro es *"significativamente distinto (menor o mayor)"* que θ_0. También es posible interpretar el p-valor como *"la probabilidad máxima de cometer el error de rechazar la hipótesis nula cuando es cierta"*, abreviado como *"tamaño del error si se rechaza la hipótesis nula"*.

Estos contrastes de significación, originalmente desarrollados por Ronald Fisher, fueron incluidos en un esquema de toma de decisiones por Jerzy Neyman y Egon Pearson, planteando que, de no ser cierta la hipótesis nula, se debe plantear una hipótesis alternativa H_1. La decisión de qué hipótesis resulta más congruente con los datos se basa en la comparación por cociente de las verosimilitudes de la muestra bajo cada una de ellas, decidiendo el rechazo de la hipótesis nula a favor de la alternativa cuando dicho cociente es, en probabilidad, inferior a un valor prefijado, α, conocido como **nivel de significación**. Dependiendo de la estructura de las hipótesis (simples, si sólo determinan un valor del parámetro, o compuestas, si determinan más de uno; a su vez, unilaterales si los valores son todos menores, o mayores, que uno dado, o bilaterales en caso contrario) la regla de decisión del contrate resulta más o menos compleja de obtener.

Computacionalmente, dada la información muestral, es más fácil calcular el p-valor que plantear el esquema de decisión de Neyman-Pearson, por lo que es la estrategia utilizada en la práctica. Dado el carácter breve e introductorio de este capítulo, no se profundizará más en este esquema de decisión, que puede consultarse en, por ejemplo, Casella and Berger (2007), Blais (2020) o Almudevar (2021).

13.6. Inferencia estadística paramétrica sobre poblaciones normales

Como consecuenca del TCL (12.5), el supuesto de que la distribución poblacional es una normal es el caso más habitual en la práctica, siendo requisito básico en muchísimas técnicas estadísticas. En este caso, las distribuciones muestrales de los estimadores de los parámetros poblacionales, tanto de la media μ como de la varianza σ^2, son conocidas, lo que facilita la construcción de intervalos de confianza y contrastes de hipótesis.

Así, dada una distribución poblacional normal y una m.a.s. de tamaño n,

- Para estimar la varianza poblacional, σ^2, el estimador máximo verosímil es la varianza muestral (13.6), que es sesgado. El estimador insesgado es la cuasivarianza muestral (13.8). Sus distribuciones en el muestreo son:

$$\frac{nS^2}{\sigma^2} = \frac{(n-1)S_c^2}{\sigma^2} \sim \chi^2_{n-1} \tag{13.17}$$

Así, el intervalo de confianza a nivel $(1-\alpha)$ es:

$$IC_{(1-\alpha)} = \left[\frac{nS^2}{\chi^2_{n-1,\alpha/2}}, \frac{nS^2}{\chi^2_{n-1,1-\alpha/2}}\right] \tag{13.18}$$

o, equivalentemente, usando la proprocionalidad entre varianza y cuasivarianza muestrales

$$IC_{(1-\alpha)} = \left[\frac{(n-1)S_c^2}{\chi^2_{n-1,\alpha/2}}, \frac{(n-1)S_c^2}{\chi^2_{n-1,1-\alpha/2}}\right] \tag{13.19}$$

donde $\chi^2_{n-1,\alpha/2}$ representa el cuantil en la distribución.

Para el contraste de $H_0 \equiv \sigma^2 = \sigma_0^2$, la "distancia" es $\frac{nS^2}{\sigma_0^2} = \frac{(n-1)S_c^2}{\sigma_0^2}$, lo que permite calcular los p-valores mediante una distribución χ^2_{n-1}.

Para el caso de querer estimar la desviación típica, basta con calcular la raiz cuadrada del estimador de la varianza, o si se busca un intervalo de confianza, la raiz de los extremos del intervalo para la varianza. Los contrastes de hipótesis son equivalentes, ya que $\sigma^2 = \sigma_0^2 \equiv \sigma = \sigma_0$.

- Para estimar el parámetro μ se utiliza el estimador media muestral $\hat{\mu} = \bar{X}$, en el que coinciden los métodos de mínimos cuadrados, máxima verosimilitud y el de los momentos, siendo insesgado y de varianza mínima.
- Si la varianza poblacional es conocida, la distribución en el muestreo de la media muestral es:

$$\bar{X} \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right) \equiv \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \sim N(0,1) \tag{13.20}$$

  El intervalo de confianza a nivel $(1-\alpha)$ es:

$$IC_{(1-\alpha)} = \left[\bar{X} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}}, \bar{X} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right] \tag{13.21}$$

  donde $z_{\alpha/2}$ representa el cuantil en una distribución normal estándar.

Para el contraste de $H_0 \equiv \mu = \mu_0$, la "distancia" es $\frac{\bar{X}-\mu_0}{\sigma/\sqrt{n}}$, lo que permite calcular los p-valores mediante una distribución $N(\mu, \sigma)$.

- Si la varianza poblacional es desconocida, se sustituye por su estimación, por lo que la distribución en el muestreo de la media muestral es:

$$\frac{\bar{X}-\mu}{S/\sqrt{n-1}} \equiv \frac{\bar{X}-\mu}{S_c/\sqrt{n}} \sim t_{n-1} \tag{13.22}$$

  El intervalo de confianza a nivel $(1-\alpha)$ es:

$$IC_{(1-\alpha)} = \left[\bar{X} - t_{n-1,\alpha/2}\frac{S}{\sqrt{n-1}}, \bar{X} + t_{n-1,\alpha/2}\frac{S}{\sqrt{n-1}}\right] \tag{13.23}$$

o, equivalentemente,

$$IC_{(1-\alpha)} = \left[\bar{X} - t_{n-1,\alpha/2}\frac{S_c}{\sqrt{n}}, \bar{X} + t_{n-1,\alpha/2}\frac{S_c}{\sqrt{n}}\right] \tag{13.24}$$

donde $t_{n-1,\alpha/2}$ representa el cuantil de la distribución *t-Student*.

Para el contraste de $H_0 \equiv \mu = \mu_0$, la "distancia" es $\frac{\bar{X}-\mu_0}{S/\sqrt{n-1}} = \frac{\bar{X}-\mu_0}{S_c/\sqrt{n}}$, lo que permite calcular los p-valores mediante una distribución t_{n-1}.

Cuando el interés es la comparación de dos poblaciones normales independientes, X e Y, a partir de muestras $(X_1, ..., X_n)$ y $(Y_1, ..., Y_m)$, se modifican los estimadores.

- Para la comparación de las varianzas poblacionales, la distribución muestral del cociente es una distribución F-Snedecor:

$$\frac{\frac{mS_Y^2}{(m-1)\sigma_Y^2}}{\frac{nS_X^2}{(n-1)\sigma_X^2}} \equiv \frac{S_{cY}^2/\sigma_Y^2}{S_{cX}^2/\sigma_X^2} \sim F_{m-1,n-1} \tag{13.25}$$

  que permite calcular intervalos de confianza de forma idéntica a la expuesta a los casos anteriores pero con la distribución F. Un caso muy frecuente es querer contrastar si ambas varianzas poblacionales son iguales (el cociente entre ellas es la unidad).

- Para la comparación de las medias poblacionales, el caso más común es asumir que las varianzas (aunque desconocidas) son iguales, por lo que el estimador es:

$$\frac{(\bar{X}-\bar{Y}) - (\mu_X - \mu_Y)}{\sqrt{\frac{nS_X^2+mS_Y^2}{n+m-2}}\sqrt{\frac{1}{n}+\frac{1}{m}}} \sim t_{n+m-2} \tag{13.26}$$

  si se utiliza la varianza muestral o

$$\frac{(\bar{X}-\bar{Y}) - (\mu_X - \mu_Y)}{\sqrt{\frac{(n-1)S_{cX}^2+(m-1)S_{cY}^2}{n+m-2}}\sqrt{\frac{1}{n}+\frac{1}{m}}} \sim t_{n+m-2} \tag{13.27}$$

  si se utiliza la cuasivarianza muestral.

Al ser distribuciones t-Student, los intervalos de confianza y contrastes de hipótesis son similares a los del caso de una única población con las correcciones pertinentes.

13.7. Inferencia sobre poblaciones normales con R

Los datos sobre calidad del aire en la ciudad de Nueva York (`airquality`) recogen la variable `Wind` que mide, en mph, la velocidad del viento entre el día 1 de mayo y el 30 de septiembre. Se particionan dichos datos en dos variables X = Velocidad del viento hasta el 15 de julio e Y = Velocidad del viento desde el 16 de julio. Asumiendo que las distribuciones poblacionales son normales, se propone:

a) Determinar una estimación de la velocidad media y de la desviación típica de ambas variables, usando el método de máxima verosimilitud.

```
library(fitdistrplus)
library(DescTools)
x <- airquality$Wind[1:76]
y <- airquality$Wind[77:153] # Se particiona la muestra en los dos períodos
mle_x <- fitdist(x, distr = "norm", method = "mle")
mle_y <- fitdist(y, distr = "norm", method = "mle")
mle_x
#> Fitting of the distribution ' norm ' by maximum likelihood
#> Parameters:
#>       estimate Std. Error
#> mean 10.640789  0.4274723
#> sd    3.726618  0.3022685
mle_y
#> Fitting of the distribution ' norm ' by maximum likelihood
#> Parameters:
#>      estimate Std. Error
#> mean 9.283117  0.3581657
#> sd   3.142891  0.2532613
```

Es resultado muestra como estimaciones la media y la desviación típica muestrales respectivamente, junto al *"error estandar"*, o desviación típica de los estimadores respectivos.

La orden `plot(mle_x)` permite un análisis gráfico de las estimaciones realizadas; por ejemplo si optamos por el primer período, se obtiene la fig.13.2 siguiente:

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

b) Establecer un intervalo de confianza para la velocidad media del viento hasta el 15 de julio, con un nivel de confianza del 95 %.

```
MeanCI(x, conf.level = 0.95)
#>      mean    lwr.ci    upr.ci
#> 10.640789  9.783563 11.498016
```

c) Calcular un intervalo de confianza para la desviación típica de la velocidad del viento desde el 16 de julio, con un nivel de confianza del 90 %.

Figura 13.2: Resultados gráficos de la estimación por máxima verosimilitud

```
sqrt(VarCI(y, conf.level = 0.9))
#>      var   lwr.ci   upr.ci
#> 3.163501 2.795147 3.655468
```

d) ¿Se puede considerar que las varianzas poblacionales en ambos períodos son iguales, con un nivel de significación del 1 %?.

```
VarTest(x, y, conf.level = 0.99, alternative = "two.sided")
#>
#>  F test to compare two variances
#>
#> data:  x and y
#> F = 1.4062, num df = 75, denom df = 76, p-value = 0.1406
#> alternative hypothesis: true ratio of variances is not equal to 1
#> 99 percent confidence interval:
#>  0.7727136 2.5616496
#> sample estimates:
#> ratio of variances
#>           1.406197
```

El estadístico de contraste, *F-Snedecor*, arroja un valor de 1.4062, con un p-valor de 0.1406. Como este p-valor no es pequeño, no hay suficiente evidencia empírica como para rechazar la hipótesis nula de igualdad de varianzas.

e) Teniendo en cuenta los resultados del apartado anterior, ¿se puede afirmar que la velocidad media del viento en el primer período es mayor que la del segundo, con un nivel de significación del 1 %?

```
t.test(x, y, conf.level = 0.99, alternative = "greater", var.equal = TRUE)
#>
#>  Two Sample t-test
#>
#> data:  x and y
#> t = 2.4212, df = 151, p-value = 0.008328
#> alternative hypothesis: true difference in means is greater than 0
#> 99 percent confidence interval:
#>  0.03918338        Inf
#> sample estimates:
#> mean of x mean of y
#> 10.640789  9.283117
```

Un p-valor tan bajo (0.008) indica que existe suficiente evidencia empírica como para rechazar la hipótesis nula de igualdad de medias, por lo que sí se podría afirmar, con un nivel de confianza del 99 %, que la velocidad media del viento en el primer período es superior a la del segundo.

13.8. Inferencia estadística no paramétrica: contrastes de normalidad

Hasta ahora, se ha supuesto que la distribución muestral del estimador era "funcionalmente" conocida, aunque dependiente de un parámetro (o varios). Sin embargo, hay situaciones donde no se conoce cómo se distribuyen los datos, debiendo decidir qué distribución los ha generado. Es lo que se conoce como *"inferencia estadística no paramétrica"*. En este capítulo no se aborda un planteamiento sistemático de esta rama, sino que se presenta la situación más habitual en la práctica, que es decidir si se puede mantener que una muestra proviene de una distribución normal, supuesto básico en muchas técnicas estadísticas.

Posiblemente el test más potente para contrastar la normalidad sea la prueba de Shapiro-Wilks, que asume como hipótesis nula que los datos están generados por una distribución normal. Un rechazo de esta hipótesis (p-valor muy bajo) debería hacer reflexionar sobre la adecuación de muchas técnicas y la interpretación de los resultados. En **R**, la función `shapiro.test()` proporciona dicho contraste de normalidad.

Una alternativa es el uso del test de Kolmogorov-Smirnov, diseñado para comparar las distribuciones de dos muestras, fijando que una de ellas sea la distribución normal (este test puede ser igualmente utilizado para cualquier otra distribución usual). La función `ks.test()` permite en **R** obtener los resultados de este contraste.

Como ejemplificación del uso del test de Shapiro-Wilk en **R**, utilizando los datos sobre calidad del aire en la ciudad de Nueva York del ejemplo anterior (13.7), se va a contrastar si se puede asumir que las variables *Temp* y *Wind* están generadas por distribuciones normales:

```
shapiro.test(airquality$Temp)
#>
#>  Shapiro-Wilk normality test
#>
#> data:  airquality$Temp
#> W = 0.97617, p-value = 0.009319
shapiro.test(airquality$Wind)
#>
#>  Shapiro-Wilk normality test
#>
#> data:  airquality$Wind
#> W = 0.98575, p-value = 0.1178
```

Para la variable *Temp* el p-valor es muy bajo (0.0093), por lo que hay suficiente evidencia empírica como para rechazar que provenga de una distribución normal. Por el contrario, en el caso de la variable *Wind*, el p-valor (0.1178) no es pequeño, por lo que no hay suficiente evidencia como para rechazar que esté generada por una distribución normal. La fig. 13.3 siguiente muestra la comparación entre los cuantiles empíricos de ambas variables y los teóricos de una distribución normal.

Figura 13.3: Q-Q Plots normales para las variables Temp (izq) y Wind (der)

RESUMEN
La inferencia estadística permite estimar la distribución poblacional de una variable a partir de la información suministrada por una muestra. A partir de estimadores muestrales, se plantean los métodos de estimación puntual, construcción de intervalos de confianza y resolución de contrastes de significación sobre diversas hipótesis.
Para el caso de poblaciones normales, se han desarrollado las expresiones operativas de los métodos anteriores. Igualmente, se ha presentado un contraste de normalidad que permite decidir cuándo el supuesto de normalidad es adecuado o no.

Capítulo 14

Muestreo y remuestreo

Mª Leticia Meseguer Santamaría y *Manuel Vargas Vargas*

14.1. Introducción al muestreo

Muchas investigaciones científicas abordan el estudio de características de un conjunto de elementos, que se denomina *"población"*. Sin embargo, no siempre es posible estudiar la totalidad del colectivo, por problemas de accesibilidad, confidencialidad, costes económicos o temporales, etc. En estos casos, se recurre a extraer un subconjunto de la población que se pretende que sea representativo de ésta respecto a las características estudiadas. El objetivo es obtener información relevante que pueda extrapolarse al total de la población, proceso denominado genéricamente "muestreo".

En otros casos, la conveniencia de muestrear una población está relacionada con la credibilidad de los resultados obtenidos o de las propiedades de los modelos utilizados. Así, muchas técnicas cuantitativas dividen la información disponible en un subconjunto de *"entrenamiento"* o *"estimación"* y otro de *"contraste"* o *"validación"*; una elección inadecuada puede alterar los resultados e invalidar las conclusiones. Por último, caso muy común en ciencias sociales, no se puede acceder a la medición directa de los fenómenos, por lo que se accede a ella mediante encuestas, que precisa de una metodología de muestreo rigurosa y diseñada previamente.

Para fijar terminología, se define **población** como el conjunto de casos de interés a los que se quiere generalizar los resultados de la investigación, denominados genéricamente **individuos**; a veces, sólo es posible acceder a una parte de la población, por lo que se utiliza el término **población objetivo** para el conjunto total y **población muestreable** al conjunto al que se tiene acceso. Por coherencia, cuando ambos colectivos no coincidan, los resultados deben extrapolarse sólo al último de ellos.

Las características de interés de la población se consideran *"variables"*, en el sentido estadístico del término, por lo que son estudiadas mediante su distribución. En algunos casos, ésta será completamente desconocida, siendo de interés su determinación completa; en otros muchos, se

buscará determinar sólo algunos aspectos (medidas de posición, dispersión, etc.) o los parámetros que rigen una distribución funcionalmente conocida. En todo caso, el objetivo del estudio es determinar la distribución estadística de estas variables, conocida como *"distribución poblacional"*; a veces, por simplicidad en el lenguaje, a esta distribución o, incluso, a las variables, se les denomina población (por ejemplo, no es infrecuente encontrar enunciados del tipo *"la población sigue una distribución binomial"*, identificando población con variable, o *"la población es una distribución normal"*, identificando población con distribución).

Se define **muestra** como un subconjunto de la población, que será utilizado para caracterizar la distribución poblacional y **unidad muestral** a cada individuo de la muestra. Si la muestra tiene la misma distribución que la población, se dice que es **representativa**, mientras que en caso contrario se denomina **sesgada**.

Siempre será preferible un método de muestreo que proporcione muestras representativas, característica ligada a la forma de seleccionar la muestra y al tamaño de ésta. Así, siempre que sea posible, se recomienda utilizar un **muestreo probabilístico**, basado en la selección aleatoria de la muestra (conociéndose, por tanto, la probabilidad de que cada individuo salga seleccionado), lo que permite extender los resultados a toda la población, cuantificando posibles sesgos y detallando un error máximo dentro de un nivel de confianza seleccionado al inicio del proceso.

No siempre será posible utilizar un muestreo probabilístico, por lo que existe una colección de métodos de muestreo no probabilístico (conveniencia, bola de nieve, cuotas, etc.), que no se detallarán en este capítulo. La característica común a todos ellos es que los resultados obtenidos de la muestra no se deben extrapolar a la población, ya que no está garantizada la representatividad.

En el resto del capítulo se presentarán brevemente los métodos de muestreo probabilístico más usuales. Para mayor detalle, se pueden consultar las referencias Chaudhuri and Stenger (2005), Arnab (2017) o Wu and Thompson (2020).

14.2. Muestreo aleatorio simple

El método básico de muestreo es el conocido como **muestreo aleatorio simple (m.a.s.)**, ya introducido en el capítulo anterior (epígrafe 13.2) consistente en seleccionar totalmente al azar a los individuos de la muestra, por lo que todos tienen la misma probabilidad de formar parte de ella. Si cada individuo sólo puede aparecer una vez en la muestra, se habla de **muestreo sin reemplazamiento**, mientras que se denomina **muestreo con reemplazamiento** en caso contrario.

Este procedimiento es el que se asume en la inmensa mayoría de las técnicas estadísticas convencionales, pero presenta dos inconvenientes. En primer lugar, presupone que existe un registro, o listado, completo de todos los individuos de la población (lo que no siempre es posible), y puede resultar costosa (en medios, tiempo y dinero) su aplicación práctica. En segundo lugar, presupone que la característica estudiada es homogénea en todos los individuos de la población, es decir, la distribución poblacional es idéntica en todos los individuos. Frecuentemente, esta homogeneidad poblacional no se cumple, por lo que sería necesario abordar otros métodos de

muestreo que se expondrán más adelante. En todo caso, si se utiliza un m.a.s., la heterogeneidad induce un aumento de la variabilidad muestral, hecho que debe ser tenido en cuenta en la interpretación de resultados.

Además de la forma de selección, el factor que determina la representatividad de una muestra es su tamaño (véase el teorema de Glivenko-Cantelli del capítulo anterior ((13.3)). Al utilizar la información muestral para aproximar las características poblacionales se comete el llamado **error muestral**; si el tamaño de la muestra está determinado, este margen de error marca el grado de precisión con el que se pueden extrapolar los resultados. Una alternativa es predeterminar un error muestral a cierto nivel de confianza y calcular cuál es el menor tamaño muestral que cumple ese requisito.

El error muestral, ϵ, depende de la característica poblacional que se quiera analizar (frecuentemente, la media, el total poblacional o la proporción), del estimador utilizado y del nivel de confianza. Si se asume una distribución poblacional normal, la expresión general sería:

$$\epsilon_\alpha = z_{1-\alpha/2}\sigma(\hat{\theta}) \tag{14.1}$$

expresión que es frecuentemente extrapolada a distribuciones poblacionales no normales.

Para el caso de estimar la media poblacional utilizando la media muestral, sustituyendo en la ecuación anterior (14.1), la relación entre error muestral, nivel de confianza y tamaño muestral sería:

$$\epsilon_\alpha = z_{1-\alpha/2}\sqrt{(1-\frac{n}{N})\frac{S^2}{n}} \tag{14.2}$$

Operando y despejando el tamaño muestral, se obtiene la expresión:

$$n = \frac{z^2_{1-\alpha/2}NS^2}{N\epsilon^2_\alpha + z^2_{1-\alpha/2}S^2} \tag{14.3}$$

En algunos casos se considera que se está muestreando una población de tamaño infinito, lo que produce una simplificación de la fórmula de obtención del tamaño muestral:

$$\epsilon_\alpha = z_{1-\alpha/2}\sqrt{\frac{S^2}{n}} \Longrightarrow n = \frac{z^2_{1-\alpha/2}S^2}{\epsilon^2_\alpha} \tag{14.4}$$

Si se está interesado en estimar el total poblacional, un procedimiento análogo conduce a las ecuaciones:

$$\epsilon_\alpha = z_{1-\alpha/2}\sqrt{N^2(1-\frac{n}{N})\frac{S^2}{n}} \Longrightarrow n = \frac{z^2_{1-\alpha/2}N^2S^2}{\epsilon^2_\alpha + z^2_{1-\alpha/2}NS^2} \tag{14.5}$$

Por último, si se desea estimar la proporción poblacional, P, de individuos que cumplen algún criterio, se puede particularizar el caso del estimador de la media poblacional sobre una población binomial, por lo que el resultado obtenido es:

$$n = \frac{z^2_{1-\alpha/2}NPQ}{(N-1)\epsilon^2_\alpha + z^2_{1-\alpha/2}PQ} \tag{14.6}$$

siendo $Q = 1 - P$. En la práctica es muy frecuente que se desconozca la varianza poblacional, por lo que se suele recurrir a alguna estimación previa, con lo que se tiene una aproximación al tamaño muestral requerido.

Para ejemplificar el proceso de obtención de una muestra aleatoria simple en **R**, se usará la librería *samplingbook* y el conjunto de datos "iris", correspondiente a las medidas, en centímetros, de largo y ancho de los sépalos y pétalos de 150 flores, equidistribuidas entre las especias *setosa*, *versicolor* y *virginica* (además de ilustrar este epígrafe, también servirá de ejemplo para otros tipos de muestreo, 14.3).

```
library(samplingbook)
datos_ej <- data.frame(iris)
```

En este caso, por simplicidad, se considerará que la población es el conjunto de las 150 flores disponibles, y que se desea una muestra aleatoria simple sin reemplazamiento para determinar la longitud media de los sépalos con un error de 0.3 centímetros al 95 % de confianza. La función *"sample.size.mean" permite calcular el tamaño de muestra necesario para cumplir estos requisitos.

```
sd <- sd(datos_ej$Sepal.Length) # Se considera como la desviación típica
↪  poblacional
N <- nrow(datos_ej) # Tamaño de la población
e <- 0.3 # Margen de error prefijado
sample.size.mean(e, sd, N, level = 0.95)
#>
#> sample.size.mean object: Sample size for mean estimate
#> With finite population correction: N=150, precision e=0.3 and standard
↪  deviation S=0.8281
#>
#> Sample size needed: 25
```

Así, basta con una muestra aleatoria simple de tamaño 25 para poder estimar la longitud media de los sépalos con los requisitos dados. Para obtener la muestra concreta, la función *"sample"* proporciona los valores obtenidos (conjunto de 25 valores aleatorios entre 1 y N=150) y permite seleccionar los casos que conforma la muestra:

```
set.seed(196) # Fija la semilla de aleatorización para poder reproducir los
↪  resultados
muestra <- sample(1:N, 25, replace = FALSE) # Si se quisiera un muestreo con
↪  reemplazo, se utilizaría la sentencia replace=TRUE
datos_muestra <- datos_ej[muestra, ] # Se seleccionan los datos que conforman
↪  la muestra
head(datos_muestra)
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 122          5.6         2.8          4.9         2.0  virginica
```

```
#> 133          6.4          2.8          5.6          2.2  virginica
#> 104          6.3          2.9          5.6          1.8  virginica
#> 95           5.6          2.7          4.2          1.3 versicolor
#> 147          6.3          2.5          5.0          1.9  virginica
#> 73           6.3          2.5          4.9          1.5 versicolor
```

Para finalizar, usando la función *"Smean"* se puede obtener la estimación de la media poblacional, así como su error estándar y un intervalo de confianza. Aunque en la práctica el valor poblacional es desconocido, en este ejemplo sí se puede obtener a partir del conjunto de todos los datos, lo que permite comparar la estimación con el verdadero valor buscado.

```
Smean(datos_muestra$Sepal.Length, N, level = 0.95)
#>
#> Smean object: Sample mean estimate
#> With finite population correction: N=150
#>
#> Mean estimate: 5.952
#> Standard error: 0.1145
#> 95% confidence interval: [5.7275,6.1765]
mean(datos_ej$Sepal.Length) # Valor de la media poblacional
#> [1] 5.843333
```

En este ejemplo, el error cometido sería de $5{,}952 - 5{,}843 = 0{,}109$ cm.

Si interesa estimar el total poblacional, basta con multiplicar la estimación de la media por el tamaño poblacional, N. Por último, si se desea estimar una proporción poblacional, el proceso sería idéntico al descrito, pero usando la función *Sprop*.

14.3. Muestreo estratificado

Como se ha comentado en el epígrafe anterior, uno de los supuestos del muestreo aleatorio simple es que todos los elementos de la población tienen la misma distribución poblacional. Sin embargo, es frecuente que haya *"grupos"* de individuos que presenten características diferenciadas en alguna variable respecto a la característica que se quiere estudiar.

En el ejemplo anterior, se ha muestreado para estimar la longitud media de los sépalos de las 150 flores recogidas en los datos. Sin embargo, al calcular la media y la desviación típica agrupando según la especie:

```
library(plyr)
estratos <- ddply(datos_ej, .(Species), summarize, media.sl =
↪  mean(Sepal.Length), desv.sl = sd(Sepal.Length))
estratos
```

```
#>      Species media.sl   desv.sl
#> 1     setosa    5.006 0.3524897
#> 2 versicolor    5.936 0.5161711
#> 3  virginica    6.588 0.6358796
```

puede observarse que las tres especies no se comportan igual respecto al parámetro de interés (longitud media de los sépalos).La especie *setosa* presenta unos valores menores de longitud media desviación típica, mientras que la especie *virginica* presenta los valores más elevados.

Si no se considerara la especie, un muestreo aleatorio simple podría sesgar los resultados, por ejemplo, con un predominio de las setosas o de las virginicas. En todo caso, la variabilidad del conjunto de datos es más elevada que en cualquiera de las especies, pues a la variación dentro de cada especie se une la variación entre especies. Este hecho hace aumentar el tamaño muestral necesario para estimar con un margen de erro prefijado.

En general, cuando existen grupos de individuos con un comportamiento más homogéneo dentro del grupo y diferenciado entre grupos, no resulta apropiado aplicar un m.a.s. En estos casos, es recomendable el denominado **muestreo estratificado**, donde se realiza previamente una partición de la población en **estratos** y se selecciona una m.a.s. dentro de cada grupo.

La estratificación presenta ventajas, como el aumento de la representatividad de la muestra (se necesita un menor tamaño muestral que en el m.a.s.), la reducción del error muestral (la variabilidad es menor en cada estrato) y el incremento de probabilidad de representación en la muestra de grupos con características diferenciadas. Por el contrario, no siempre resulta evidente la relación entre la variable de estratificación y las de interés.

Una de las decisiones que se han de tomar en el muestreo estratificado es el reparto de tamaño muestral entre los distintos estratos, procedimiento conocido como **afijación**. Las dos opciones más utilizadas son la **afijación proporcional**, que reparte el tamaño muestral en función de los tamaños poblacionales de cada estrato, y la **afijación óptima**, que considera también los diferentes valores de la variabilidad dentro de cada estrato.

Para ejemplificar el proceso de muestreo estratificado en el caso de la base de datos utilizada, se procede a considerar cada especie de iris como un estrato, ya que se ha comprobado que presentan distintas distribuciones poblacionales respecto a la variable *longitud del sépalo*. Como en el epigrafe anterior, se quiere estimar la longitud media de la variable con un margen de error de 0.3 al 95 % de confianza. Dentro de la librería *samplingbook* se puede utilizar la función *stratasize* para determinar el tamaño muestral que cumple estos requisitos

```
stratasize(e, Nh = c(50, 50, 50), Sh = estratos[, 3], level = 0.95)
#>
#> stratamean object: Stratified sample size determination
#>
#> type of sample: prop
#>
#> total sample size determinated: 11
```

Como se aprecia, para garantizar al 95 % de confianza un margen de error de 0.3 cm es necesario un tamaño muestral de 11, sensiblemente inferior al requerido con un m.a.s. (25). Una vez determinado el tamaño, el criterio de afijación elegido distribuye la muestra entre los estratos.

```
stratasamp(
  n = 11, Nh = c(50, 50, 50), Sh = estratos[, 3], type = "prop"
)
#>
#> Stratum 1 2 3
#> Size    4 4 4
stratasamp(
  n = 11, Nh = c(50, 50, 50), Sh = estratos[, 3], type = "opt"
)
#>
#> Stratum 1 2 3
#> Size    3 4 5
```

Como los tres estratos tienen el mismo tamaño poblacional, la afijación proporcional distribuye la muestra equitativamente; sin embargo, la afijación óptima, al considerar las diferencias en variabilidad, asigna más muestra al estrato con mayor variabilidad y menos muestra al de menor variabilidad (como los tamaños muestrales son necesariamente números enteros, se puede producir una ligera diferencia entre el tamaño muestral calculado globalmente y la suma de los tamaños de cada estrato).

Con la afijación óptima estimada, se procede a la selección de la submuestra en cada estrato (m.a.s.) y a la obtención de los datos que conforman la muestra.

```
set.seed(196) # Fija la semilla de aleatorización
muestra1 <- sample(1:50, 3, replace = FALSE)
muestra2 <- sample(51:100, 4, replace = FALSE)
muestra3 <- sample(101:150, 5, replace = FALSE) # m.a.s. en cada estrato
muestra_estr <- c(muestra1, muestra2, muestra3)
datos_muestra_estr <- datos_ej[muestra_estr, ] # Selección de los datos que
↪  conforman la muestra
datos_muestra_estr
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 5            5.0         3.6          1.4         0.2     setosa
#> 40           5.1         3.4          1.5         0.2     setosa
#> 31           4.8         3.1          1.6         0.2     setosa
#> 90           5.5         2.5          4.0         1.3 versicolor
#> 81           5.5         2.4          3.8         1.1 versicolor
#> 59           6.6         2.9          4.6         1.3 versicolor
#> 58           4.9         2.4          3.3         1.0 versicolor
#> 137          6.3         3.4          5.6         2.4  virginica
#> 136          7.7         3.0          6.1         2.3  virginica
#> 132          7.9         3.8          6.4         2.0  virginica
```

```
#> 107          4.9          2.5          4.5          1.7  virginica
#> 121          6.9          3.2          5.7          2.3  virginica
```

Finalmente, usando la función *"Smean"* se obtiene la estimación de la media poblacional, así como su error estándar y un intervalo de confianza, al igual que se hizo con el m.a.s.

```
Smean(datos_muestra_estr$Sepal.Length, N, level = 0.95)
#>
#> Smean object: Sample mean estimate
#> With finite population correction: N=150
#>
#> Mean estimate: 5.925
#> Standard error: 0.3108
#> 95% confidence interval: [5.3158,6.5342]
```

14.4. Otros tipos de muestreo probabilístico

Existen otros métodos de muestreo probabilístico que buscan simplificar la extracción de una muestra representativa, entre los que destacan el **muestreo por conglomerados** y el **muestreo sistemático**.

Cuando la población es muy grande, es frecuente que se puedan establecer (o construir a partir de alguna variable) subgrupos, o **clusters**, que tengan las mismas características que todo el conjunto respecto a la variable de interés. En esos casos, a efectos de estimación, sería equivalente muestrear toda la población o sólo un cluster, con el consiguiente ahorro de tamaño muestral, tiempo y coste. Es el conocido como **muestreo por conglomerados**.

Por ejemplo, se puede estar interesado en estimar el tiempo medio que el alumnado de E.S.O. dedica a estudiar matemáticas en España. Obtener una muestra para todo el pais puede ser costoso en tiempo, recursos materiales y tamaño muestral; sin embargo, se puede asumir que no existen diferencias entre provincias respecto a esta variable, por lo que sería posible muestrear sólo en una provincia (o pocas). En este caso, el muestreo por conglomerados consistiría en una primera etapa de selección aleatoria de clusters (provincia/s en este ejemplo) y, posteriormente, aplicar un método de muestreo sobre dicha selección (que, a su vez, podría ser un m.a.s. o un muestreo estratificado).

No conviene confundir los conceptos de estrato y cluster, aunque ambos sean subgrupos de la población total. En el primer caso, los individuos de cada estrato son muy homogéneos entre sí y diferenciados del resto de estratos. En el segundo caso, los individuos de cada cluster tienen la misma variabilidad que el conjunto de la población, no habiendo diferencias entre clusters respecto a la variable de interés. Así, la ganancia en el muestreo estratificado proviene de trabajar con menores variabilidades intra-estratos, mientras que en el muestreo por conglomerados proviene de utilizar una subpoblación más pequeña.

En otras situaciones, si se dispone de un marco poblacional (listado completo de los individuos), es posible plantear un mecanismo sencillo de obtención de la muestra. Si se tiene un tamaño

poblacional N y se quiere una muestra de tamaño n, se pueden establecer $k = N/n$ bloques, elegir al azar un número entre 1 y k (que permite seleccionar el primer elemento de la muestra) y, a partir de esa posición, dar saltos de magnitud k en el listado para seleccionar el resto de unidades muestrales. Es el método conocido como **muestreo sistemático** o, más técnicamente, **muestreo sistemático uniforme de paso k**.

Como ejemplo, supóngase que se quiere obtener una muestra de 50 individuos en una población de 2000. El paso sería $k = 2000/50 = 40$ unidades y se selecciona aleatoriamente una unidad entre las 40 primeras (supóngase que la número 13); el resto de la muestra se obtendría sumándole el paso a la primera seleccionada (13, 53, 93, 133, 173, y así hasta la 1973).

Por último, los métodos expuestos no son incompatibles, sino que se pueden combinar por etapas, dando lugar a los conocidos como **muestreos polietápicos"**. Por ejemplo, la Encuesta de Población Activa, elaborada por el Instituto Nacional de Estadística, adopta un muestreo bietápico, en primer lugar, estratificado entre secciones censales y, en segundo lugar, muestreando entre las viviendas familiares de cada sección.

14.5. Técnicas de remuestreo: Bootstrap.

Cuando se infiere una característica poblacional a partir de una muestra, sólo se dispone del valor concreto que el estadístico toma sobre dicha muestra. Salvo en raras ocasiones, no se dispone de su distribución en el muestreo, o sólo se tiene una aproximación asintótica, por lo que no se pueden evaluar sus propiedades estadísticas con tamaños muestrales no elevados. En otros casos, es la complejidad analítica de muchas técnicas actuales de análisis de datos la que dificulta la determinación de la distribución de las estimaciones de los parámetros.

En estos casos, el método **bootstrap** propone sustituir la distribución poblacional (desconocida) por una estimación conocida (como puede ser la distribución empírica o una aproximación paramétrica) que, mediante remuestreo, sirva para generar muestras aleatorias a partir de la muestra original. Se obtiene así una distribución de remuestreo, llamada también **distribución bootstrap**, cuyo comportamiento sobre la estimación aproxima a la de la distribución muestral en torno al parámetro, lo que permite evaluar la precisión de las estimaciones.

El método bootstrap más sencillo, llamado **bootstrap uniforme** o **bootstrap naïve**, parte de la aproximación de la distribución poblacional por la distribución empírica de la muestra. Supóngase que se tiene una muestra $X = (x_1, ..., x_n)$ que es utilizada para obtener un estimador $T(X) = \hat{\theta}$ para un parámetro poblacional θ. Su distribución empírica (véase (13.2)) es:

$$\hat{\mathbb{F}}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{(-\infty,x]}(X_i) \tag{14.7}$$

donde, para cada valor x, se obtiene la proporción de elementos de la muestra que toman un valor menor o igual que x. Con esta distribución,

- Se genera una primera muestra $X^{*1} = (x_1^{*1}, ..., x_n^{*1})$, obtenida mediante muestreo aleatorio simple con reemplazamiento de la muestra original, que permite evaluar es estadístico $T^{*1}(X^{*1}) = \hat{\theta^{*1}}$.

- Siguiendo el mismo procedimiento, se puede generar un número elevado (B) de muestras bootstrap, $X^{*1}, ..., X^{*B}$, que permiten obtener el valor del estadístico sobre cada una de ellas $T^{*1}(X^{*1}), ..., T^{*B}(X^{*B})$
- Con estos valores, se obtiene la distribución bootstrap. Por ejemplo, se puede utilizar esta distribución para calcular la media bootstrap del estadístico $T(X)$, $\bar{T}^* = \frac{1}{B}\sum_{b=1}^{B} T^*(X^{*b})$, y cuyo error estándar es:

$$\hat{S}e_{boot} = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(T(X^{*b}) - \bar{T}^*\right)^2} \tag{14.8}$$

  expresión que aproxima al error del estadístico $T(X)$ para estimar θ.

Ejemplo: Los datos sobre calidad del aire en la ciudad de Nueva York (*airquality*) recogen la variable *"Temp"* que mide la temperatura, en grados Fahreheit, entre el día 1 de mayo y el 30 de septiembre. Como se ha visto en el capítulo anterior, epígrafe 13.7, no se puede asumir que dicha variable esté generada por una distribución normal, por lo que no se podrían utilizar los intervalos de confianza mostrados en el epígrafe 13.6. El objetivo es calcular una estimación de la temperatura media y dar un intervalo al 95 % de confianza.

```
media <- mean(airquality$Temp)
```

En este ejemplo, se utiliza la muestra dara obtener un valor del estadístico *media muestral* ($\bar{X} = 77{,}88$), estimador insesgado de la media poblacional. Sin embargo, al no conocer la distribución en el muestreo (no se asume ningún tipo de distribución poblacional ni se puede hacer uso de aproximaciones asintóticas), no se podría construir un intervalo de confianza.

Aplicando el método bootstrap (en su versión uniforme), se van a obtener 5000 muestras de tamaño 20, mediante remuestreo con reemplazamiento,

```
set.seed(196) # Se fija la semilla para permitir la reproducibilidad
B <- 5000 # Se fija el número de remuestras
muestras_boot <- numeric(B) # Se almacenan todos los valores del estadístico
for (k in 1:B) {
  remuestra <- sample(airquality$Temp, 20, replace = TRUE)
  muestras_boot[k] <- mean(remuestra)
}
media_boot <- mean(muestras_boot)
media_boot # Media de los 5000 valores medios de las remuestras
#> [1] 77.87309
desv_boot <- sd(muestras_boot)
desv_boot # Desviación típica de los 5000 valores medios de las remuestras
#> [1] 2.090255
```

Para construir los intervalos de confianza, se calculan los valores críticos al 95 % de confianza a partir de las remuestras tipificadas y los extremos del intervalo:

```
tip_boot <- (muestras_boot - media_boot) / desv_boot # Remuestras tipificadas
val_crit <- quantile(tip_boot, c(0.025, 0.975))
val_crit # Valores críticos
#>      2.5%     97.5%
#> -1.972530  1.950437
ic_inf <- media_boot - val_crit[2] * desv_boot
ic_sup <- media_boot - val_crit[1] * desv_boot
ic_boot <- c(ic_inf, ic_sup)
ic_boot
#>    97.5%     2.5%
#> 73.79618 81.99618
```

Así, con el método bootstrap, no es necesario asumir ninguna distribución en el muestreo. Aún así, la representación gráfica de la distribución empírica de los valores medios remuestreados y los extremos del intervalo de confianza, muestra cómo la distribución de las remuestras sobre el estadístico se asemeja a la distribución del estadístico sobre el parámetro.

Figura 14.1: Distribución empírica de la media remuestreada

La libreria *"boot"* de R permite también obtener réplicas de un estadístico sobre una muestra. La función básica de esta librería es *boot*, que permite utilizar distintos métodos de remuestreo. En su estructura más simple, basta con indicar los datos originales, el estadístico que se quiere remuestrear y el número de réplicas. Así, si se quiere estimar, por ejemplo, la mediana de la población con 1000 remuestras, se puede recurrir a la función *boot*:

```
library(boot)
estadistico <- function(data, i) {
  median(data[i]) # Se especifica aquí el estadístico remuestreado
}
set.seed(196)
```

```
mediana_boot <- boot(airquality$Temp, estadistico, R = 1000)
mediana_boot
#>
#> ORDINARY NONPARAMETRIC BOOTSTRAP
#>
#>
#> Call:
#> boot(data = airquality$Temp, statistic = estadistico, R = 1000)
#>
#>
#> Bootstrap Statistics :
#>     original  bias    std. error
#> t1*       79  -0.052    1.071655
```

Como resultado, se obtiene el valor del estadístico sobre la muestra original, el sesgo estimado y el error estándar. Para calcular los intervalos de confianza, basta con utilizar la función *boot.ci* sobre la muestra bootstrap obtenida, indicando el nivel de confianza y el tipo de intervalo (por defecto, *"all"* proporciona todos los intervalos disponibles; en el ejemplo, se usa el método de los percentiles)

```
boot.ci(mediana_boot, conf = 0.95, type = "perc")
#> BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
#> Based on 1000 bootstrap replicates
#>
#> CALL :
#> boot.ci(boot.out = mediana_boot, conf = 0.95, type = "perc")
#>
#> Intervals :
#> Level     Percentile
#> 95%   (77, 81 )
#> Calculations and Intervals on Original Scale
```

La función *boot* permite modificaciones del bootstrap uniforme mediante parámetros adicionales. Por ejemplo, el parámetro *strata* se utiliza para generar remuestreos estratificados cuando la muestra original también lo es. Aunque no se han comentado dado el carácter introductorio de este capítulo, existen otros métodos bootstrap, que se pueden obtener especificándolos mediante el parámetro *sim*. Por defecto, el valor es *"ordinary"*, que corresponde al bootstrap uniforme; otras alternativas pueden ser *"parametric"* para bootstrap paramétrico, *"balanced"*, *"permutation"* o *"antithetic"* para otros métodos más avanzados.

RESUMEN

El muestreo probabilístico busca seleccionar una muestra representativa de una población, que permita inferir la distribución poblacional o alguno de sus parámetros. Las decisiones básicas para un correcto proceso de muestreo son el método utilizado (aleatorio simple, estratificado, polietápico, etc.), que depende de la estructura de la población, y la determinación del tamaño muestral que garantice el margen de error asumible. La técnica **bootstrap** de remuestreo permite aproximar la distribución de estadísticos muestrales sin asumir ninguna hipótesis sobre la distribución poblacional, ventaja muy útil para evaluar la precisión de los estimadores en muchísimas técnicas complejas.

Parte IV

Modelización estadística

Capítulo 15

Modelización lineal

Víctor Casero y María Durbán

15.1. Modelización

Se acude a modelos de regresión para intentar explicar la relación entre dos o más variables. Para ello se predefine un modelo que pretende explicar el comportamiento de la variable respuesta o dependiente, denotada por Y, utilizando la información proporcionada por las variables explicativas (independientes, predictoras... denotadas por $X_1, \ldots, X_k$). Pero dichas variables pueden ser de muy distinto tipo. Si la variable respuesta es continua, más concretamente, si se puede asumir que sigue una distribución normal, y al menos una de las variables explicativas es también continua, se puede acudir a la modelización lineal que se desarrolla en este capítulo. Sin embargo, si la variable respuesta fuese de otro tipo, por ejemplo, dicotómica, la modelización lineal explicada en este Capítulo no sería adecuada. En el Capítulo 17 del modelo lineal generalizado quedará más clara esta distinción.

El primer paso en este proceso de modelización es intentar explicar una variable respuesta, que a partir de aquí se supone continua y con distribución normal, a partir de una sola de las variable explicativas, de forma *lineal*[1]. Dicho modelo probablemente no será bueno, no explicará bien la variable respuesta, esto es lo lógico si la realidad que se pretende explicar es compleja, pero podría ser suficiente para el propósito del estudio[2]. Por ejemplo, se sabe que el peso de una persona está relacionado con muchos factores, pero uno de los más determinantes es la altura. Si se recogen datos de pesos y alturas de un conjunto de personas se puede ajustar el modelo y obtener una explicación "suficiente", aunque parcial, del peso de una persona a partir de su altura. Es claro que la inclusión de otras variables en el modelo puede ayudar a "explicar" mejor la variable respuesta. Se llega así al denominado *modelo de regresión lineal múltiple* que

[1] La relación entre las variables viene determinada por una combinación *lineal* de los parámetros. En contraposición se encuentran los modelos *no lineales*, como por ejemplo: $Y = \beta_0 \cdot e^{\beta_1 X_1}$.

[2] Ayudará a ello obtener el coeficiente de *bondad de ajuste* (se verá en la Sección 15.2.1).

se puede expresar matemáticamente como:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \ldots + \beta_k X_{ki} + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2), \tag{15.1}$$

donde

- β_0 es el *término independiente* o *constante del modelo*,
- $\beta_1, \ldots, \beta_k$ son los *coeficientes de regresión* o *parámetros* del modelo, que se obtendrán a partir del conocimiento de los datos observados $(x_{1i}, \ldots, x_{k_i}, y_i)$ y reflejan la magnitud del efecto *lineal* que las variables explicativas X_i tienen sobre la variable explicada Y,
- y ϵ_i es el término de error del modelo, lo que no es capaz de explicar la parte determinista del mismo: $\beta_0 + \beta_1 x_{1i} + \ldots + \beta_k x_{ki}$, y que se supone sigue una distribución normal de probabilidad, con media 0 y varianza constante σ^2.
- Además se asume que las observaciones son independientes.

Siguiendo con el ejemplo del peso, añadir alguna variable genética, o el sexo, ayudará a mejorar la "bondad" del modelo lineal. Otro ejemplo consiste en pretender explicar el salario en un determinado sector económico en función de... años de experiencia, formación, situación familiar, sexo, etc. Nótese que entre las variables explicativas sí puede haber variables de distinto tipo, continuas, categóricas, etc[3]. Ahora bien, la interpretación de los coeficientes dependerá del tipo de variable al que va asociado, como se verá en los casos prácticos (Sección 15.4).

15.2. Procedimiento de modelización

15.2.1. Estimación del modelo

Los datos recogidos u observados sirven para **especificar** la relación predefinida de antemano, mediante la **estimación** de los coeficientes β_i que mejor ajustan dicha relación, utilizando el método de *mínimos cuadrados*. Además, se hacen los correspondientes contrastes para decidir si cada coeficiente es **significativamente distinto de 0**, esto es, tiene un *efecto* significativo sobre la respuesta[4] por lo que tiene sentido mantenerlo en el modelo. En la práctica el coeficiente es *significativo* si su **p-valor** asociado es lo suficientemente pequeño[5]. Si no lo es, es oportuna la eliminación de dicha variable explicativa del modelo. En tal caso se vuelven a estimar los coeficientes de las variables que se mantienen hasta llegar a un modelo con todos los coeficientes significativos, iterando las veces necesarias. No obstante se han desarrollado métodos automáticos de selección de variables, basados en la comparación de la varianza residual

[3] Conviene mencionar la estrecha relación entre regresión lineal múltiple y ANOVA. Ésta última técnica es un caso particular de la primera, concretamente cuando todas las variables explicativas son categóricas (véase la Sección **??**).

[4] desde el punto de vista estadístico *la influencia/efecto sobre la respuesta no es fruto del azar*.

[5] Se acepta mayoritariamente un p-valor inferior a 0.05, lo que supone un nivel de confianza de las estimaciones del 95 %. Pero dicho valor es arbitrario. Podría considerarse un nivel de confianza del 99 % y entonces sería significativo si el p-valor fuese inferior a 0.01. En general, cuanto menor sea el p-valor más confianza se tendrá en las conclusiones. Si está próximo, aunque sea menor de 0.05 ó 0.01, hay que tomar las conclusiones con reservas, igual que si es ligeramente superior.

haciendo uso del test F, mediante el estadístico AIC (*Criterio de Información de Akaike*), etc. [6]. Junto con los contrastes se pueden obtener los intervalos de confianza de los coeficientes, que si son significativos no contendrán el valor 0.

A la par del contraste de significación de cada coeficiente se obtiene el contraste de significación global del modelo. La hipótesis nula es que todos los coeficientes $\beta_1, \ldots, \beta_k$ son 0, dicho de otro modo, que el conocimiento de las variables $X_1, \ldots, X_k$ no aporta mejor información para explicar los valores de Y que su simple media de las respuestas.

También se ha de obtener la **bondad del ajuste** del modelo, típicamente proporcionando el coeficiente de determinación (corregido), R^2 (adimensional, que toma valores entre 0 y 1). Cuanto mejor explique el modelo los datos muestrales, más próximo a 1 será el R^2, indicando el 1 una linealidad perfecta de los datos. Por el contrario, un R^2 cercano a 0 indica un mal ajuste de los datos utilizando el modelo de regresión estimado.

Es habitual valorar conjuntamente la significación global del modelo, su bondad de ajuste y la significación de cada uno de los coeficientes, considerando apropiados aquellos modelos que son *significativos* y tienen la "suficiente" *bondad*, aunque tengan coeficientes no significativos.

15.2.2. Validación del modelo

Aunque el modelo sea significativo se debe *validar*, es decir, comprobar los supuestos estadísticos que subyacen al modelo. Para ello se utilizan los **residuos** del modelo, lo que queda sin explicar por la regresión estimada, la diferencia entre el valor observado y el estimado. Matemáticamente,

$$e_i = Y_i - \hat{Y}_i = Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \ldots + \hat{\beta}_k X_{ki})$$

Los supuestos a comprobar son:

- Los residuos han de tener **varianza constante** (tienen por definición **media cero**).
- Los residuos han de seguir la **distribución** de probabilidad **Normal**.
- Las observaciones tienen que ser **independientes**.
- La **relación** entre la variable respuesta y las explicativas se supone **lineal**[7].
- Las variables explicativas son linealmente independientes: ninguna puede ser explicada como combinación lineal de las otras. Si se diese el caso se tendría el conocido problema de la *multicolinealidad* y debería quitarse del modelo la variable explicada por el resto.

15.2.3. Interpretación de los coeficientes

Una vez validado el modelo se pasan a interpretar los coeficientes significativos. La regla general de interpretación de cada coeficiente de regresión estimado $\hat{\beta}_i$ es, el cambio *medio* en el valor de la variable respuesta...

[6] Se puede consultar el Capítulo 10 de Faraway (2002)

[7] Podrían estudiarse relaciones no lineales de las variables, pero lineales en los parámetros, como: $Y = \beta_0 + \beta_1 X + \beta_2 X^2$.

- por cada cambio unitario de la variable explicativa, si es *continua*.
- al cambiar el nivel de la variable explicativa, desde uno de los niveles que se toma como referencia, si la variable es *categórica*.

... y *ceteris paribus*, esto es, manteniendo constante el valor de las demás variables.

Habrá que tener en cuenta las magnitudes de cada variable, porque la influencia real en la respuesta podría ser de poca magnitud (quizá por las unidades o escala utilizada), pero significativa estadísticamente.

15.2.4. Predicción

La utilidad del modelo estimado (especificado) queda plasmada en su utilización para **predecir** nuevos valores, $\hat{Y}_i$, a partir del conocimiento/asignación de los valores de las variables explicativas, $X_1, \ldots, X_k$. No obstante dichas predicciones corresponden al valor *medio*, como se ha indicado, y se pueden dar sus correspondiente intervalos de confianza, tanto para predicciones individuales como para predicciones de la media.

15.3. Procedimiento con R: la función `lm`

R tiene implementada la función `lm` para ajustar/estimar modelos de regresión lineal múltiple.

```
lm(formula, data = ..., ...)
```

El argumento mínimo necesario es `formula` donde se predeterminará la relación entre las variables respuesta y explicativas de una forma bastante intuitiva:

- `Y ~ X`, es la fórmula a utilizar para definir un modelo *simple* donde `Y` denota la variable respuesta y `X` denota la variable explicativa: $Y = \beta_0 + \beta_1 X$.
- `Y ~ X1 + X2`, define un modelo lineal *múltiple* con 2 variables explicativas: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$.
- `Y ~ X1 + X2 + X3 - 1` elimina del modelo lineal múltiple (de 3 variables) el término independiente, β_0.
- ...

Si las variables no se encuentran en el `Environment` se le debe especificar en `data =` el conjunto de datos donde se encuentran.

Hay que hacer notar que `R` considera, por defecto, las variables explicativas como cuantitativas. Si se tienen variables categóricas codificadas con números hay que indicarle que la trate como categórica, usando la función `factor`[8]. De no hacerlo, el procedimiento la consideraría numérica, con el consecuente error de interpretación del coeficiente asociado.

[8] Si no está ya definida como `factor` en el conjunto de datos.

A partir de los datos disponibles de `Y, X1, ...`, mediante el *método de mínimos cuadrados*, la función `lm` estimará los coeficientes β_i asociados a cada variable `Xi`, calculará sus errores estándar con el que obtiene sus estadísticos de contraste (de la *t de Student*)[9] y su significación. En el objeto "lm" que se genera también almacena valores ajustados, residuos, etc. que se pueden mostrar a través de funciones genéricas disponibles en `R`. Algunas de ellas son:

- `print`: muestra un resumen corto
- `summary`: resumen completo
- `coef`: estimaciones de los coeficientes del modelo
- `confint`: intervalos de confianza de los coeficientes
- `fitted.values`: valores ajustados del modelo (para cada observación del `data.frame`)
- `residuals`: residuos del modelo.

15.4. Casos prácticos

En esta sección se van a utilizar los datos `airquality`[10] que consisten en 154 medidas (de 6 variables) de calidad del aire en Nueva York. Las variables consideradas aquí son:

- `Ozone`: Media de ozono (en ppb, partes por billón)
- `Solar.R`: Radiación solar (en Lagleys)
- `Wind`: Velocidad media del viento (en millas por hora)
- `Temp`: Temperatura máxima diaria (en grados Fahrenheit)

El objetivo es establecer la relación entre la cantidad de ozono en la atmósfera, variable respuesta, y las variables meteorológicas `Solar.R`, `Wind` y `Temp`, variables explicativas. Los valores disponibles de las cuatro variables permiten considerarlas como variables continuas.

Antes de proceder con el ajuste múltiple se pueden realizar los ajustes simples, individuales. La Figura 15.1 representa 3 regresiones lineales simples, de cada una de las 3 variables explicativas frente a la variable respuesta `Ozone`. Se visualiza en cada gráfico un *diagrama de dispersión* de sólo dos variables, la explicativa en el eje X y la respuesta en el eje Y, obteniéndose la popularmente denominada *nube de puntos*. En dicho gráfico se puede ver si entre las variables hay relación positiva/directa (si a mayores valores de X se observan mayores valores de Y) o negativa/inversa y si tienen o no relación lineal. En cada gráfico se ha añadido la correspondiente recta de regresión lineal (con su *intervalo de confianza*), que podría no ser la más apropiada, como parece que ocurre entre `Ozone` y `Wind` y entre `Ozone` y `Temp` cuya relación parece más bien no lineal, aunque suficiente (en función del interés del estudio). El código para el obtener el primer gráfico sería:

```
library(ggplot2)
ggplot(airquality, aes(Solar.R, Ozone)) +
  geom_point() +
```

[9] Con los errores estándar también se pueden obtener los intervalos de confianza de los coeficientes
[10] Conjunto de datos incluido con la instalación `base` de `R`. Ejecútese ?airquality

Figura 15.1: Gráficos de dispersión de las variables explicativas frente a la variable respuesta.

```
  theme(aspect.ratio = 1) +
  geom_smooth(method = "lm")
```

En regresión lineal múltiple no es posible visualizar en un sólo gráfico la relación entre la variable Y y varias variables explicativas X_i, salvo si sólo se tienen 2 explicativas que se tendría un gráfico en 3 dimensiones, generalmente difíciles de visualizar.

15.4.1. Estimación de los coeficientes

Se comienza ajustando el modelo lineal para la variable `Ozone`, intentando explicarla a partir de las variables continuas `Solar.R`, `Wind` y `Temp`. El modelo de regresión lineal múltiple se expresaría matemáticamente como:

$$Ozone = \beta_0 + \beta_1 Solar.R + \beta_2 Wind + \beta_3 Temp + \epsilon$$

Más adelante se introducirá una variable categórica para enriquecer el análisis. La definición en `R` del modelo se puede ver como primer argumento de la función `lm`. El objeto que genera la función `lm` se guarda bajo el nombre de `airq.lm`, y a continuación se muestra su resumen con `summary`:

```
airq.lm <- lm(Ozone ~ Solar.R + Wind + Temp, data = airquality)
summary(airq.lm)
#>
#> Call:
#> lm(formula = Ozone ~ Solar.R + Wind + Temp, data = airquality)
#>
```

```
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -40.485 -14.219  -3.551  10.097  95.619
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -64.34208   23.05472  -2.791  0.00623 **
#> Solar.R       0.05982    0.02319   2.580  0.01124 *
#> Wind         -3.33359    0.65441  -5.094 1.52e-06 ***
#> Temp          1.65209    0.25353   6.516 2.42e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 21.18 on 107 degrees of freedom
#>   (42 observations deleted due to missingness)
#> Multiple R-squared:  0.6059, Adjusted R-squared:  0.5948
#> F-statistic: 54.83 on 3 and 107 DF,  p-value: < 2.2e-16
```

En la salida del `summary` se proporcionan las estimaciones de los coeficientes del modelo (columna `Estimate`). El término independiente aparece como `(Intercept)` y toma el valor $\beta_0 =$ -64.3421, el coeficiente asociado a `Solar.R` es $\beta_1 = 0.0598$, etc. También aparecen sus p-valores asociados (columna `Pr(>|t|)`), los 4 coeficientes son significativos al 5 %. Interpretando la leyenda `Signif. codes`, a mayor número de asteriscos que aparecen, mayor significación del coeficiente (menor p-valor). Así, los coeficientes de `Temp` y `Wind` son más significativos que el de `Solar.R`.

También se pueden apreciar en la salida (penúltima línea) dos medidas de la *bondad del ajuste* del modelo considerado: el R cuadrado múltiple y el R cuadrado ajustado. Para comparar entre diferentes modelos, se utiliza el R cuadrado ajustado/corregido que tiene en cuenta la composición/complejidad del modelo (número de variables, etc.). Para el ejemplo el R^2(ajustado) es 0.5948, que se podría considerar suficiente (en función del objetivo del estudio), pero está claro que el modelo no explica suficientemente bien la cantidad de ozono.

Y en la última linea de la salida aparece información sobre el contraste global del modelo: Estadístico de la F, grados de libertad y p-valor asociado. Como se aprecia, el modelo es globalmente significativo (p-valor del orden de 10^{-16}).

15.4.2. Validación

Lo anterior carece de *validez* si no se satisfacen las hipótesis del modelo mencionadas arriba, principalmente varianza constante (homocedasticidad) y normalidad. Para ello se realiza el análisis de residuos. R proporciona un análisis gráfico:

```
library(ggfortify)
autoplot(airq.lm) +
  theme_minimal()
```

Figura 15.2: Gráficos de residuos

Por un lado el gráfico de residuos frente a valores ajustados (fitted) muestra cierta heterocedasticidad (varianza cambiante con el valor en el eje X) y no linealidad (ya apreciable de forma individual en la Figura 15.1). Por su parte, el gráfico Q-Q de cuantiles, que enfrenta los de los residuos frente a los de la distribución normal, muestra que los residuos presentan desviaciones de la normalidad en ambas colas.

Para completar el análisis gráfico se puede acudir a contrastes de hipótesis. El más habitual para contrastar normalidad es el de Shapiro-Wilk implementado en `R` con la función `shapiro.test`[11]. Para contrastar la homocedasticidad se puede acudir a alguno de los tres implementados para tal fin en el paquete `lmtest`: el de Breusch-Pagan `bptest`, el de Goldfeld-Quandt `gqtest` o el de Harrison-McCabe `hmctest`.

```
shapiro.test(airq.lm$residuals)
#>
#>  Shapiro-Wilk normality test
#>
#> data:  airq.lm$residuals
#> W = 0.91709, p-value = 3.618e-06
lmtest::bptest(airq.lm)
#>
#>  studentized Breusch-Pagan test
#>
```

[11] Se pueden encontrar otros contrastes de normalidad en el paquete `nortest`.

```
#> data:  airq.lm
#> BP = 5.0554, df = 3, p-value = 0.1678
```

Figura 15.3: Histogramas de las variables 'Ozone' y 'log(Ozone)'

El contraste de homocedasticidad lleva a no rechazar tal supuesto (p-valor > 0.05), pero el contraste de Shapiro-Wilk confirma la falta de normalidad (p-valor < 0.05). A este respecto, en la Figura 15.3 se muestra el histograma de la variable `Ozone` apreciándose que los datos recogidos presentan asimetría incompatible con la normalidad, asumida por defecto para la variable respuesta. Una posible solución sería el uso de una transformación logarítmica que produce cierta simetría en la variable, y por tanto más cercanía a la normalidad.

Para el análisis de colinealidad se pueden representar gráficos 2 a 2 de las variables explicativas, para visualizar lo correlacionadas que están (Figura 15.4). Pero es una análisis parcial, una de las variables explicativas podría venir explicada por el resto o varias de ellas. En este caso es mejor acudir al cálculo de los *factores de inflación de la varianza*, que se pueden obtener mediante la función `vif` del paquete `car`.

```
library(GGally)
ggpairs(airquality[, 2:4])
```

```
car::vif(airq.lm)
#>  Solar.R     Wind     Temp
#> 1.095253 1.329070 1.431367
```

Los gráficos de dispersión muestran ausencia de correlación entre `Solar.R` y `Wind` mientras que la correlación entre `Wind` y `Temp` no parece despreciable, como es palpable en la práctica. Sin embargo los VIF muestran valores inferiores a 5 (valor de referencia marcado habitualmente), por lo que el modelo no presenta multicolinealidad.

Figura 15.4: Gráfico de dispersión por pares de las variables explicativas

15.4.3. Interpretación de los coeficientes

Teniendo en cuenta la expresión del modelo de regresión lineal múltiple (15.1) la interpretación de cada uno de los coeficientes es simple y directa: lo que aumenta/disminuye la respuesta *media* en magnitud por un aumento de 1 unidad en la variable (continua) asociada al coeficiente, y *ceteris paribus*. Así,

- si el valor de la variable `Temp` aumenta en 1 unidad (1 grado Fahrenheit), independientemente del valor de `Wind` y `Solar.R`[12]), la respuesta media (cantidad media de ozono, en ppb) cambia en magnitud lo que indica su coeficiente: 1.6521, en este caso aumenta por ser positivo el coeficiente.
- al ser el de `Wind` negativo implica que aumentos de `Wind` reducen la cantidad media de ozono. La reducción por cada unidad de `Wind` (mph) es el valor de su coeficiente: -3.3336 (ppb).
- la influencia de `Solar.R` es positiva como `Temp` pero de mucha menor magnitud (por las unidades de una y otra).

Como se puede ver el impacto sobre la cantidad *media* de ozono depende de la variable y de la magnitud del coeficiente.

Conviene mencionar que las interpretaciones del modelo no deben extrapolarse a valores fuera del rango que toman las variables explicativas, porque en esas regiones podrían darse otros efectos distintos del lineal que presupone el modelo estimado.

[12] si estos cambian tendrán su correspondiente impacto en la cantidad media de ozono.

15.4.4. Predicción

Aunque el modelo no sea el más adecuado se ilustra en esta sección cómo se pueden obtener predicciones a partir de un modelo estimado con la función `predict`. Se deben asignar los valores de interés a las variables explicativas del modelo, con formato `data.frame`. Y se podrán obtener predicciones del valor medio o valores individuales de la variable respuesta, junto con sus intervalos de confianza, según se proporcione al argumento `interval` los valores `confidence` o `prediction`, respectivamente. En el siguiente ejemplo se obtienen predicciones de cantidad de ozono para un par de casos elegidos arbitrariamente (el primero corresponde a `Solar.R`=50, `Wind`=5 y `Temp`=62):

```
nueva.meteo <- data.frame(
  Solar.R = c(50, 300),
  Wind = c(5, 17),
  Temp = c(62, 90)
)
predict(airq.lm, newdata = nueva.meteo, interval = "confidence")
#>        fit      lwr      upr
#> 1 24.41075 11.01412 37.80739
#> 2 45.62141 31.46838 59.77444
predict(airq.lm, newdata = nueva.meteo, interval = "prediction")
#>        fit        lwr      upr
#> 1 24.41075 -19.662967 68.48448
#> 2 45.62141   1.311914 89.93090
```

Como se puede observar, la predicción puntual (`fit`) es la misma, pero con `prediction` es el valor predicho para un día específico con esas condiciones meteorológicas, mientras que con `confidence` es el valor medio esperado para días con esas condiciones. Por ello las predicciones de valores individuales (`prediction`) tienen un intervalo de confianza mayor que las predicciones de valores medios (`confidence`).

15.4.5. Nuevo ajuste con `log(Ozone)`

Ante los problemas de falta de normalidad de la variable `Ozone` se ajusta un nuevo modelo con la variable $\log(Ozone)$ como respuesta. Se aprovecha para introducir/definir una variable *dicotómica* para explicar su interpretación. Concretamente se usa `Temp` que se *dicotomiza* entre aquellos valores de `Temp` por encima o por debajo de su mediana:

```
mediana <- median(airquality$Temp)
Temp.f <- factor(as.numeric(airquality$Temp > mediana))
lairq.lm <- lm(log(Ozone) ~ Wind + Solar.R + Temp.f, data = airquality)
summary(lairq.lm)
#>
#> Call:
```

```
#> lm(formula = log(Ozone) ~ Wind + Solar.R + Temp.f, data = airquality)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.55347 -0.29689  0.02409  0.37171  1.18373
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  3.3879872  0.2232099  15.178  < 2e-16 ***
#> Wind        -0.0885666  0.0161038  -5.500 2.61e-07 ***
#> Solar.R      0.0030723  0.0005973   5.143 1.23e-06 ***
#> Temp.f1      0.6999123  0.1158384   6.042 2.25e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.5572 on 107 degrees of freedom
#>   (42 observations deleted due to missingness)
#> Multiple R-squared:  0.5972, Adjusted R-squared:  0.5859
#> F-statistic: 52.89 on 3 and 107 DF,  p-value: < 2.2e-16
```

Como se ve, las estimaciones de los coeficientes cambian respecto al modelo anterior, al cambiar la variable respuesta y una variable explicativa del modelo. Salen todos los coeficientes significativos (al 5 %), el modelo global también significativo y el R^2(corregido) es similar al del modelo anterior. En el Capítulo 18 de modelos aditivos generalizados se verá como se puede modelar la relación entre `Ozone` y el resto de variables de una forma más satisfactoria. No obstante, este segundo modelo puede ser útil para ilustrar la relación entre las variables, sin olvidar que se ha de comprobar su validez. Para ello se haría de nuevo el análisis de residuos (que se deja de tarea al lector, al obtenerse de manera idéntica al anterior). En los gráficos de residuos se observará mayor homocedasticidad, linealidad y normalidad que en el caso anterior.

15.4.6. Coeficientes de variables categóricas

En este apartado se aborda la interpretación de coeficientes asociados a variables categóricas. `Temp.f` toma valores numéricos 0 y 1, según la temperatura sea menor o mayor a la mediana respectivamente. En la salida anterior de `R` sólo aparece `Temp.f1`. El 1 final indica que el coeficiente está asociado a la categoría 1 de `Temp.f`. Para los cálculos con estas variables categóricas `R` toma una categoría como referencia (la primera "alfabéticamente", si no se especifica expresamente) y presenta un coeficiente para cada una de las restantes categorías representando el cambio desde la categoría de referencia a cada una de ellas (técnicamente utiliza variables *dummys*). La categoría de referencia está considerada en los cálculos del término independiente del modelo. Por lo tanto, el coeficiente de `Temp.f1` indica que, *ceteris paribus*, la cantidad media de ozono de días con temperaturas por encima de la mediana (categoría 1) es 0.6999 ppb mayor que los días con temperaturas inferiores a la mediana (categoría 0).

15.4.6.1. Comparativa: Regresión frente a ANOVA

En la Figura 15.1 se pueden apreciar regresiones simples *puras* (variable continua frente a variable continua). Si en el gráfico se enfrentan la variable `Ozone` con la variable categórica `Temp.f` no se tendrá una visualización similar[13]. En realidad, al incluir un *factor* se está realizando un contraste de la *t de Student* para averiguar si existen diferencias entre cada categoría y la de referencia. Técnicamente tales contrastes dos a dos son equivalentes al contraste ANOVA, aunque éste permite comparar si todas las categorías son iguales, o no. No obstante, el gráfico ayuda a la interpretación del coeficiente pues permite comparar las medias de cada categoría, viendo si aumenta o disminuye de una categoría a otra.

15.5. Comentarios finales

Una buena referencia para ampliar conocimientos sobre este tema utilizando `R` es Faraway (2002). En capítulos posteriores se abordarán modelizaciones más complejas, como por ejemplo, los modelos lineales generalizados, GLM, (Capítulo 17), los modelos aditivos generalizados, GAM, (Capítulo 18) y los modelos mixtos (Capítulo 19). También se verán modelos *sparse* y métodos penalizados de regresión (Capítulo 20) como la regresión *ridge* que permite manejar la presencia de multicolinealidad.

Queda fuera de este capítulo el tema de la confusión de variables, esto es, aquellas variables que aun no siendo de interés directo en el análisis se quieren obtener resultados independientemente de sus valores. El ejemplo típico es la variable sexo, esto es, se quiere obtener la influencia de una variable X_i en la respuesta Y, independientemente del sexo del individuo. La solución ya ha salido: se incluye la variable en el modelo para que *ceteris paribus* se puede interpretar en el análisis.

También podrían haberse considerado interacciones entre variables, que se suelen interpretar como sinergias o antagonismos. Pero dada la limitación de espacio y el carácter introductorio no se ha hecho. Pues la interpretación de dichas interacciones suele ser generalmente compleja. En el Capítulo 3 de James et al. (2013) puede encontrarse un ejemplo.

[13] `ggplot(airquality, aes(Temp.f, Ozone)) + geom_point()`.

Resumen

En este Capítulo se ha introducido el modelo de regresión lineal, en particular:

- Se ha presentado directamente el modelo de regresión lineal múltiple indicando los pasos para realizar el análisis: estimación, validación, interpretación y predicción. La regresión lineal simple es un caso particular que se puede obtener a partir de lo comentado en el Capítulo.
- Se ha mostrado el uso de `R` para el ajuste de este tipo de modelos.
- Se han presentado diversos casos prácticos, que sirven para ilustrar la interpretación de coeficientes, tanto asociados a variables continuas como a categóricas, la interpretación de las predicciones y el resto de análisis.
- Se han mencionado distintos problemas de modelización que el análisis ayuda a detectar proponiendo a su vez soluciones para solventarlos.

Capítulo 16

Modelos lineales generalizados

Víctor Casero y María Durbán

16.1. Motivación

Como se ha mencionado en el Capítulo 16 de modelos lineales, el objetivo detrás del uso de modelos es el de intentar explicar una variable en función de otras que se cree que influyen en ella. Por ejemplo, podría interesar *predecir*

- si un empleado abandonará la empresa, o no, en función de los años de experiencia, la formación, etc. O si un paciente sufrirá o no un accidente coronario en función de su edad, sexo, nivel de colesterol, etc.
- el número de días que el empleado estará de baja laboral en función de su antigüedad, salario, etc. O el número de días que un paciente estará en el hospital en función del motivo de ingreso, su edad, sexo, etc.

Estos casos no pueden analizarse (correctamente) con el modelo de regresión lineal múltiple visto en el Capítulo 16, por que la *variable respuesta*, la que interesa predecir en cada ejemplo, no sigue una distribución de probabilidad normal (supuesto necesario para utilizar la regresión lineal). Concretamente,

- abandonar (o no) la empresa o sufrir (o no) un accidente coronario se puede modelizar mediante una variable *dicotómica/binaria*, típicamente asignando los valores 0 y 1, lo que encaja perfectamente con una distribución de probabilidad de Bernoulli (muy distinta de la normal).
- el número de días... de baja, de hospitalización... es una variable de tipo *recuento* (sólo valores positivos) modelizable mediante una variable discreta que podría seguir una distribución de Poisson (que por ser discreta, es también distinta a la normal, aunque podría parecerse).

En este Capítulo se mostrará cómo el **modelo lineal generalizado** es el modelo teórico adecuado que permite abordar el problema de regresión con *variables respuesta con distinto tipo de distribución de probabilidad*. Concretamente permitirá explicar una *función de la media* a partir de los distintos valores que pueden tomar las variables explicativas. Este Capítulo se centra en la regresión logística y la regresión de Poisson, casos particulares de este modelo, que permiten dar respuesta (correcta) a los dos casos planteados. Un buen libro de referencia para este Capítulo y otros es James et al. (2013).

16.2. Modelo y sus componentes

El **Modelo Lineal Generalizado**, **GLM**[1] se puede escribir como:

$$\mu = g^{-1}(\eta)$$

en el que se tienen los siguientes componentes:

1. $\mu = E(Y)$, el componente aleatorio: la *media* de la variable respuesta Y, que puede seguir cualquier distribución de probabilidad de la familia exponencial. Entre ellas están las más habituales: la normal (por tanto el modelo de regresión lineal es un caso particular de GLM), la Bernoulli/binomial (la utilizada en la regresión logística), la Poisson, la gamma, etc.

2. $\eta = X\beta$, el componente sistemático, el *predictor lineal*, la 'estructura' que aportan las variables explicativas/predictoras $X = (X_1, \ldots, X_k)$, que intentan explicar el comportamiento de la variable respuesta, donde $\beta = (\beta_1, \ldots, \beta_k)$ es el vector de coeficientes (parámetros) a estimar del modelo.

3. $g(\cdot)$, la novedad de los GLM, la denominada **función de enlace**, que relaciona los dos componentes anteriores. Esta función puede tomar distintas formas como se ve en el próximo apartado.

Por lo tanto, gracias al GLM se podrá intentar explicar la relación entre una variable respuesta, y una o varias variables que pueden 'explicarla'. El procedimiento de análisis suele constar de dos partes:

1. La **especificación** de la relación o estructura predefinida de antemano, mediante la *estimación* de los *coeficientes* que mejor ajustan dicha relación, utilizando el método de *máxima verosimilitud*[2]. Más adelante se verá cómo se interpretan tales coeficientes y cómo se puede comprobar la adecuación del modelo.
2. La utilización del modelo estimado (especificado) para **predecir** nuevas respuestas, según sea el caso: valores, probabilidades de ocurrencia, etc.

[1] Generalized Linear Model. *Cuidado*: *general* y *generalizado* no son sinónimos en este contexto.
[2] A diferencia de la regresión lineal múltiple, que utiliza el de *mínimos cuadrados*.

Para la correcta aplicación de los GLM es crucial la elección de dichas variables respuesta y explicativas (que podrían ser de distinto tipo: numéricas -continuas o discretas- o categóricas/cualitativas -dicotómicas o politómicas-[3]) y la distribución de probabilidad más apropiada para la respuesta. Como muestra sirva percatarse de que una misma variable podría ser explicativa o respuesta, por ejemplo 'diabetes' (sí o no), según se tenga interés en explicar la influencia de la diabetes en otra variable, o se pretende estudiar la influencia de otras variables en la diabetes. También una misma variable podría considerarse y utilizarse como variable de distinto tipo, por ejemplo 'edad', como variable discreta (*años cumplidos*) o como categórica ordinal (*grupos de edad*), aunque es una variable continua (puede tomar cualquier valor en un intervalo).

16.2.1. Función enlace

Cada distribución de probabilidad tiene asociada una función de enlace *canónica*[4]:

- Para la normal es la identidad: $g(\mu) = \mu = \eta$.
- Para la Bernoulli, es la función logit: $g(\mu) = logit(\mu) = log(\mu/(1-\mu)) = \eta$.
- Para la Poisson, es el logaritmo: $g(\mu) = log(\mu) = \eta$.
- Para la Gamma es la inversa: $g(\mu) = 1/\mu = \eta, \ldots$

Tanto en el caso de la regresión logística como la de Poisson aparece el logaritmo (neperiano) en la función de enlace, lo que conduce a efectos **multiplicativos** de los *factores* o *covariables* X_i sobre la respuesta, como se verá más claramente en la sección 16.3.3. Este es un punto que las distingue de la regresión lineal, en la que los efectos son **aditivos**.

16.2.2. GLMs en R

En el paquete `stats` (de la distribución 'base' de `R`) se encuentra la función `glm` que se utiliza para obtener el ajuste:

```
glm(formula, family = ..., data = ..., ...)
```

- `formula`: que refleje el *predictor lineal*: `Y ~ X1 + X2 + X3`, etc.
- `family`: la distribución aleatoria de la variable respuesta: `gaussian`, `binomial`, `poisson` ... que determina la función de enlace (`binomial -> logit`, etc. `?family` o `?glm` para más detalles).

Las 'herramientas' utilizadas para `lm` se pueden usar para `glm` (aunque algunas interpretaciones varían). Así se pueden usar `summary` para detectar los predictores importantes, obtener los valores ajustados con `fitted`, etc.

[3] En este contexto, las variables categóricas se denominan *factores* y las numéricas *covariables*.
[4] Podrían considerarse otras para cada distribución, pero excede el nivel de este Capítulo.

16.3. Regresión logística

La regresión logística es el caso más 'famoso' de GLM, de gran relevancia en distintos contextos: Medicina, Economía, etc. Se utiliza cuando la *variable respuesta* es *dicotómica*, del tipo pertenencia, o no, a un determinado grupo (fumadores, enfermedad X, parados, morosos, ...). Habitualmente se considera que toma el valor $Y = 1$ si se pertenece al grupo e $Y = 0$ si NO se pertenece al grupo. Tal tipo de variable se puede modelizar con una *distribución de Bernoulli* caracterizada por el parámetro p que representa la probabilidad de pertenencia al grupo.

El objetivo principal suele ser predecir la pertenencia o no al grupo de un nuevo individuo/elemento basado en la información conocida del mismo a través de las *variables explicativas*. Para ello se estimará el modelo, con los datos disponibles, para determinar que variables influyen *significativamente* en la variable respuesta[5].

¿Por qué no tiene cabida aquí el uso del modelo de regresión lineal múltiple?
Al ajustar un modelo del tipo: $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k$, las estimaciones realizadas $\hat{Y}$ serán números reales que rara vez coincidirán con 0 ó 1 (que es el valor esperable de la variable respuesta). Dicho de otro modo, si sólo se tuviese una variable explicativa, al ajustar el modelo de regresión lineal simple, la recta se saldría o no llegaría a los valores posibles de respuesta (0 ó 1), como ocurre en la Figura 16.1 (izquierda) obtenida a partir de los datos del ejemplo coronario que se manejará.

El **modelo de regresión logística múltiple** permite ajustar el modelo

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k \tag{16.1}$$

mediante el cual se pueden obtener las probabilidades $P[Y = 1] = p$ y $P[Y = 0] = 1 - p$. Para ello se utiliza una *función de enlace* que transforma el predictor lineal $\eta = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k$ en valores plausibles para una probabilidad. La función logística definida como

$$p = \frac{e^{\eta}}{1 + e^{\eta}} \tag{16.2}$$

y representada en la Figura 16.1 (derecha), es la más habitual, y de la que recibe su nombre la regresión logística.

La ventaja del modelo logístico es que la función tiende a cero por la izquierda y a uno por la derecha, por lo que la predicción será siempre un valor válido para una probabilidad. Como contrapartida, a medida que las probabilidades se acercan a cero o a uno la relación entre el predictor y la probabilidad es menos lineal, lo que complica la interpretación de los coeficientes.

16.3.1. Procedimiento de ajuste

A partir de los datos disponibles, tanto de los valores de las variables *predictoras* como de las *respuestas*, recogidas en el estudio

[5] desde el punto de vista estadístico *la influencia/efecto no es fruto del azar*.

Figura 16.1: Gráfico de dispersión del diagnóstico frente a la edad (izquierda) y función logística (derecha).

1. Se **estimarán los coeficientes** β_i del modelo, para especificar la relación entre las variables, comprobando si tales estimaciones, $\hat{\beta}_j$, son **significativas**[6].
2. Se valorará la eliminación de variables no significativas, atendiendo a la significación global del modelo, etc.
3. Se comprobará la adecuación del modelo final obtenido
4. Se interpretarán los coeficientes y quedará listo para hacer **predicciones** de probabilidades o valores para **clasificar** a los individuos/elementos.

Nota: como se ha mencionado en el Capítulo 16 del modelo lineal, para decir que un coeficiente es **significativo** se acude a su **p-valor** asociado, concretamente cuando dicho p-valor es lo suficientemente pequeño, como norma general inferior a 0.05 (aunque podrían tomarse otros valores de referencia[7]).

16.3.2. Adecuación del modelo

Para comprobar si el modelo estimado es adecuado, se compara con el modelo más simple, el que solo incluye el término independiente. Para ello se pueden utilizar distintos *contrastes basados en la deviance*[8], una medida que juega el papel de la suma de cuadrados de los residuos. En el modelo de regresión logística, la *deviance* es menos dos veces el logaritmo de la *verosimilitud*. La diferencia entre la *deviance* de un modelo más elaborado y el simple se distribuye como una chi-cuadrado, lo que permite contrastar cuál de los dos modelos ajusta mejor los datos. P-valores bajos indicarán que el modelo ajustado es inadecuado, debiendo investigarse otras

[6]Significativamente distintas de cero: tiene sentido no eliminarlas del predictor lineal.
[7]Por ejemplo, en las salidas de `R` aparecen otros tres niveles de referencia, 0.10, 0.01 y 0.001
[8]en ocasiones traducida como *Devianza*, aunque es habitual no traducirla.

posibles variables predictoras, si se incumple la hipótesis de linealidad o existe sobredispersión (por ejemplo por exceso de ceros).

Adicionalmente se puede aplicar el contraste de de la razón de verosimilitud, que también se basará en las *deviance*, las generadas al añadir cada término secuencialmente al modelo. Contrasta si cada variable (añadida en ese orden al modelo que contiene las anteriores) es significativa, lo que ayuda a decidir si mantenerla o eliminarla del modelo.

Para contrastar la bondad de ajuste del modelo es habitual en la literatura el *contraste de Hosmer-Lemeshow*. Aplicable a modelos con al menos una variable cuantitativa . Valores bajos del p-valor resultante indicarán falta de ajuste.

Respecto a las medidas del ajuste del modelo, en el modelo de regresión logística no tiene sentido calcular el coeficiente de determinación R^2, pero existen varias alternativas equivalentes para hacernos una idea de la variabilidad explicada por el modelo. Las tres medidas más habituales son la Pseudo R^2 de McFadden, la R^2 de Cox y Snell (que por construcción no puede alcanzar el 1) y la R^2 de Nagelkerke (es una corrección de la de Cox y Snell).

16.3.3. Interpretación de resultados

La interpretación de los coeficientes no es tan directa y sencilla como en el modelo lineal. A partir de las estimaciones de la regresión lineal η, modelo (16.1), se puede obtener la probabilidad de que un individuo pertenezca al grupo ($Y = 1$) utilizando la expresión (16.2). Alternativamente se pueden obtener los *odds*, con la siguiente expresión que se deduce de (16.2):

$$odds = \frac{p}{1-p} = e^{\eta} = e^{\beta_0+\beta_1 X_1+\ldots+\beta_k X_k}, \tag{16.3}$$

Los *odds* vienen a ser cuántas veces es posible pertenecer al grupo ($Y = 1$) frente a no pertenecer ($Y = 0$). Por ejemplo, si $p = 0{,}75$ el *odds* es $0{,}75/0{,}25 = 3$. Esto indica que es tres veces más posible pertenecer al grupo que no pertenecer[9].

La ecuación (16.3) se puede expresar equivalentemente como:

$$odds = e^{\beta_0} \cdot e^{\beta_1 X_1} \cdot \ldots \cdot e^{\beta_k X_k}. \tag{16.4}$$

La interpretación de los $\hat{\beta}_j$ carece de sentido, lo interpretable es $e^{\hat{\beta}_j}$, denominado **odd ratio (OR)**. Además se ve que el modelo (16.4) es *multiplicativo* (los términos aparecen multiplicando, no sumando), por lo que obtener $e^{\hat{\beta}_j}$ inferiores a 1 implican disminución de odds, y por encima de 1, implican aumento, *ceteris paribus*[10].

Con el modelo de regresión logística se puede proporcionar el llamado **riesgo relativo** (RR, *relative risk*). Es una razón de probabilidades como los *odds*, pero en vez de comparar la probabilidad de pertenecer a un grupo o a otro de la variable respuesta, compara las probabilidades de pertenecer al grupo de referencia $Y = 1$ según los valores del predictor categórico X_j. El riesgo relativo es similar al OR de la variable X_j sólo cuando la probabilidad de $Y = 1$ es baja, se suele dar como referencia que sea inferior al 10 %. Por lo tanto, OR y RR no coinciden siempre.

[9] Por esto se traduce *odds* como *ventaja*. Pero cuidado con su uso: si $Y = 1$ es tener una enfermedad, no parece muy adecuado hablar de *ventaja*.

[10] Manteniendo constante el valor de las demás variables.

16.3.4. Predicción. Curva ROC y AUC

Como se ha mencionado, el uso habitual de la regresión logística es la predicción, no tanto de valores o probabilidades sino de la clasificación en un grupo u otro de futuros individuos/elementos basado en la información conocida de ellos a través de las variables explicativas. La regla de clasificación en los grupos depende de la elección del punto de corte de la probabilidad, por ejemplo $\hat{Y} = 1$ si $\hat{p} > 0{,}7$ y $\hat{Y} = 0$ para el resto. Para ello se acude al análisis de los casos clasificados correctamente o no por el modelo ajustado: *análisis de sensibilidad y especificidad del modelo*, y su visualización más popular, la **curva ROC** (*Receiver Operating Characteristic*) y la medición de su **AUC** (*area under the curve*).

Tanto la **sensibilidad** como la **especificidad** son el porcentaje de elementos de la muestra bien clasificados por el modelo. La *sensibilidad* de los que pertenecen al grupo ($Y = 1$), la *especificidad* de los que no pertenecen ($Y = 0$). Los mal clasificados se dividen en *falsos positivos*, cuando el modelo pronostica $\hat{Y} = 1$ pero en realidad es $Y = 0$, y *falsos negativos*, el caso contrario.

Gráficamente, se pueden obtener los valores de la sensibilidad y especificidad para distintos puntos de corte (entre 0 y 1). La Figura 16.2 (izquierda) muestra este gráfico para uno de los casos prácticos[11]. Pero la forma más popular de analizar la sensibilidad y especificidad es mediante la curva ROC que representa la sensibilidad frente al ratio de falsos positivos, esto es *1-especificidad* (Figura 16.2 (derecha)). Su AUC sirve para comparar distintos modelos de regresión logística (u otros modelos con el mismo fin[12]). Cuanto mejor poder discriminante tenga el modelo más próximo a 1 será el AUC. Un clasificador aleatorio presentaría la curva ROC coincidente con la diagonal, con un AUC de 0.5.

16.4. Regresión de Poisson

Se utiliza cuando la *variable respuesta* es *discreta* y toma sólo valores positivos, modelizable con la *distribución de Poisson*. Típicamente variables de tipo *número de ...*, como el ejemplo motivador del principio: número de días de hospitalización. El parámetro que caracteriza la distribución de Poisson es λ que representa la *media* de la variable aleatoria, y a la vez la varianza (a nivel teórico).

Aquí tampoco tiene cabida el modelo de regresión lineal, principalmente porque las estimaciones realizadas $\hat{Y}$ podrían arrojar valores negativos.

De nuevo el objetivo suele ser la predicción, en el ejemplo, el número de días que un (nuevo) paciente estará ingresado, basado en la información de sus *variables explicativas*. Previo a la predicción se estimará e interpretará el modelo, a partir de los datos disponibles, para determinar las variables explicativas que influyen sobre la variable respuesta (significativas). Y de nuevo los efectos serán *multiplicativos* por ser el logaritmo su función de enlace, como se mencionó. Por ello, lo visto anteriormente para la regresión logística es asimilable a la regresión de Poisson.

[11] La interpretación del gráfico se hace en dicho apartado.
[12] Véase el Capítulo 10 de Feature selection & engineering.

16.5. Casos prácticos

16.5.1. Ejemplos de regresión logística

Se manejará el conjunto de datos `cleveland` descrito en el el paquete del libro `CDR`. Se quiere explicar la variable `diag` (dicotómica: 1, ha sufrido el accidente coronario, 0 no lo ha sufrido) a partir de otras variables.

Estimación
La variable respuesta `diag` ya aparece como factor en la base de datos, y se considera un primer modelo con variables explicativas `edad` y `dep` (variables continuas):

```
library(CDR)
reg.log <- glm(diag ~ edad + dep,
  family = "binomial", data = cleveland
)
summary(reg.log)
#>
#> Call:
#> glm(formula = diag ~ edad + dep, family = "binomial", data = cleveland)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.3804  -0.8905  -0.6210   1.0021   1.9644
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -3.08080    0.81939  -3.760  0.00017 ***
#> edad         0.03738    0.01476   2.532  0.01134 *
#> dep          0.86851    0.13791   6.298 3.02e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 417.98  on 302  degrees of freedom
#> Residual deviance: 350.64  on 300  degrees of freedom
#> AIC: 356.64
#>
#> Number of Fisher Scoring iterations: 4
```

En la salida se muestran las estimaciones de los coeficientes del modelo (columna `Estimate`) y su significatividad (columna `Pr(>|z|)`). En este ejemplo, los dos coeficientes de interés (los correspondientes a `edad` y `dep`) son significativos (al 5 % de significación).

Se completa el modelo anterior añadiendo variables *categóricas*, concretamente, `tdolor` (politómica) y `sexo` (dicotómica), para poder incluir más adelante la interpretación de los coeficientes

asociados a este tipo de variables. Para su correcta interpretación, se deben introducir en el predictor como variables de tipo `factor`[13]. Si no, el procedimiento la considera numérica, obteniendo una salida que llevaría a una interpretación errónea.

```
reg.log2 <- update(reg.log, ~ . + sexo + tdolor)
```

Adecuación del modelo

Se realiza el contraste de Hosmer-Lemeshow para comprobar la bondad de ajuste de los modelos:

```
library(ResourceSelection)
hoslem.test(reg.log$y, reg.log$fitted.values)
#>
#>  Hosmer and Lemeshow goodness of fit (GOF) test
#>
#> data:  reg.log$y, reg.log$fitted.values
#> X-squared = 5.631, df = 8, p-value = 0.6885
hoslem.test(reg.log2$y, reg.log2$fitted.values)
#>
#>  Hosmer and Lemeshow goodness of fit (GOF) test
#>
#> data:  reg.log2$y, reg.log2$fitted.values
#> X-squared = 4.8171, df = 8, p-value = 0.7769
```

Arrojan p-valores 'altos' indicando que no hay falta de ajuste.

Para realizar el test de la razón de verosimilitud con R se acude a la función `anova`:

```
anova(reg.log2, test = "Chisq")
#> Analysis of Deviance Table
#>
#> Model: binomial, link: logit
#>
#> Response: diag
#>
#> Terms added sequentially (first to last)
#>
#>
#>        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
#> NULL                     302     417.98
#> edad    1   15.447       301     402.54 8.487e-05 ***
#> dep     1   51.894       300     350.64 5.858e-13 ***
#> sexo    1   23.982       299     326.66 9.726e-07 ***
#> tdolor  3   62.153       296     264.51 2.037e-13 ***
```

[13] En este caso ya están definidas como factor en el `data.frame`.

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Utilizando la *deviance* de añadir cada término secuencialmente al modelo permite concluir que la inclusión secuencial de tales variables predictoras es significativa respecto al modelo que contiene las anteriores.

Para completar, se podría contrastar también el efecto de una variable comparando la *deviance* del modelo con dicha variable o sin ella:

```
## se elimina edad del segundo modelo
reg.log3 <- update(reg.log2, ~ . - edad)
anova(reg.log3, reg.log2, test = "Chisq")
#> Analysis of Deviance Table
#>
#> Model 1: diag ~ dep + sexo + tdolor
#> Model 2: diag ~ edad + dep + sexo + tdolor
#>   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
#> 1       297     274.45
#> 2       296     264.51  1   9.9488  0.00161 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El resultado indica que la variable `edad` se considera un predictor significativo en el modelo. Se deja al lector realizar las restantes, que le llevarán a concluir que todos son predictores significativos.

El cálculo de de la medida de bondad de ajuste de la Pseudo R^2 de McFadden se puede obtener así:

```
null <- glm(diag ~ 1, family = "binomial", data = cleveland)
1 - logLik(reg.log) / logLik(null)
#> 'log Lik.' 0.1611088 (df=3)
1 - logLik(reg.log2) / logLik(null)
#> 'log Lik.' 0.3671819 (df=7)
```

Para el primer modelo la Pseudo R^2 de McFadden es 0.16, mientras que para el segundo es 0.37.

Interpretación de los coeficientes
Se muestran los coeficientes estimados del segundo modelo, y sus correspondientes OR (sus exponenciales) que son los interpretables:

```
summary(reg.log2)
#>
#> Call:
#> glm(formula = diag ~ edad + dep + sexo + tdolor, family = "binomial",
#>     data = cleveland)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.5346  -0.6604  -0.2436   0.6568   2.3975
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -6.67669    1.30016  -5.135 2.82e-07 ***
#> edad         0.05621    0.01828   3.075   0.0021 **
#> dep          0.80890    0.16597   4.874 1.10e-06 ***
#> sexo1        1.69477    0.36801   4.605 4.12e-06 ***
#> tdolor2      0.65668    0.67357   0.975   0.3296
#> tdolor3      0.19465    0.59654   0.326   0.7442
#> tdolor4      2.58230    0.57549   4.487 7.22e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 417.98  on 302  degrees of freedom
#> Residual deviance: 264.51  on 296  degrees of freedom
#> AIC: 278.51
#>
#> Number of Fisher Scoring iterations: 5
exp(coef(reg.log2))[-1]
#>      edad       dep     sexo1   tdolor2   tdolor3   tdolor4
#>  1.057824  2.245432  5.445391  1.928371  1.214880 13.227480
```

Los coeficientes asociados a las variables *continuas* son positivos y significativos, por tanto sus OR son significativamente superiores a 1[14]. Indican que, *ceteris paribus*

- por cada año más cumplido el odds de padecer la enfermedad aumenta poco más de 1 vez, un 1.0578. ¿Y un aumento de 20 años?... En tal caso sería $e^{\beta_1 * 20} = 3.078$... En 20 años más que se triplica el odds. Ojo con la interpretación, no se triplica la probabilidad sino el odds.

[14]En regresión lineal que un coeficiente sea significativo se interpreta como significativamente distinto de cero, por el hecho de que el modelo considera efectos aditivos de las variables. El modelo de regresión logística es de efectos multiplicativos, como se ha mencionado, y por ello el valor 'neutro' es el 1. Así, un coeficiente significativo se interpreta ahora como significativamente distinto de 1.

- por un aumento de una unidad en la variable *dep* más que se duplican los *odds* del paciente, pues su OR es 2.2454.

¿Y cómo se interpretan los coeficientes asociados a las variables *categóricas*? ¿qué significa `sexo1`? ¿ `tdolor2`...? Representan *el cambio de la categoría de referencia a la mostrada.*[15]

- el de `sexo1` significa que el odds de padecer la enfermedad es más de 5 veces superior para los hombres (`sexo=1`) que para las mujeres (`sexo=0`).
- el de `tdolor4` que los de este tipo de dolor (asintomáticos) presentan un odds 13 veces superior al de los de `tdolor=1`.[16]

Nótese que no se muestra e^{β_0}, pues su interpretación carece de sentido práctico, sería el OR de las personas de 0 años, `dep = 0`, `sexo=0` y `tdolor=1`.

Predicciones
Una vez estimado e interpretado el modelo y comprobada su bondad, se puede usar para obtener predicciones. Para ello se deben asignar valores a las variables explicativas: `edad`, `dep`, `sexo` y `tdolor` (se han escogido arbitrariamente).

```
individuo <- data.frame(
  edad = 50, dep = 3,
  sexo = "1", tdolor = "4" # entre comillas por ser factores
)
```

Con la función `predict` se pueden obtener tanto el valor predicho para el predictor lineal (valor no interpretable), `eta`, como directamente la probabilidad de que dicho individuo padezca la enfermedad coronaria, `p`:

```
# (eta <- predict(reg.log2, individuo))
(p <- predict(reg.log2, individuo, type = "response"))
#>         1
#> 0.9446843
```

Con cualquiera de los valores anteriores se pueden obtener su *odds*:

```
p / (1 - p) # exp(eta)
#>        1
#> 17.07805
```

[15] `R` en las variables definidas como `factor` toma como referencia la primera categoría al ordenar los valores de la variable o bien alfabéticamente (a, b, c...) o bien numéricamente de menor a mayor. De ahí que `sexo=0` sea la categoría de referencia, porque toma los valores 0 y 1, y sólo aparezca en las estimaciones `sexo1` reflejando el cambio a tal categoría desde la de referencia. Para `tdolor`, al tomar los valores 1, 2, 3 y 4, la categoría de referencia es `tdolor=1` y aparecen coeficientes para `tdolor=2`, `tdolor=3` y `tdolor=4`.

[16] `tdolor2` y `tdolor3` no son significativos.

Es decir, un individuo con 50 años, `dep=3`, `sexo=1`(hombre) y `tdolor=4`(asintomático) tiene 17 veces más posibilidades de padecer la enfermedad que de no padecerla.

Riesgo relativo
Se ha visto que el OR para el nivel 'hombre' en el segundo modelo es 5.45. El riesgo relativo de ser hombre para un paciente de 50 años, valor `dep` de 3 y `tdolor` 4 (de nuevo valores arbitrarios) es:

```
hom <- data.frame(edad = 50, dep = 3, sexo = "1", tdolor = "4")
muj <- data.frame(edad = 50, dep = 3, sexo = "0", tdolor = "4")
(ph <- predict(reg.log2, hom, type = "response"))
#>         1
#> 0.9446843
(pm <- predict(reg.log2, muj, type = "response"))
#>         1
#> 0.7582346
ph / pm # riesgo relativo hombre/mujer
#>      1
#> 1.2459
```

El riesgo relativo de ser hombre para padecer la enfermedad cardíaca es $RR = 0{,}9447/0{,}7582 = 1{,}2459$, es decir, tiene un 24.6 % más de posibilidades de tener la enfermedad que una mujer. En este caso se observa que el OR y el RR no son valores cercanos, debido a que el diagnóstico 1 es frecuente (concretamente lo presentan el 45.9 % de los pacientes de la base de datos)

Curva ROC
El *análisis de sensibilidad y especificidad del modelo* y la curva ROC y su AUC para este ejemplo se pueden obtener con el siguiente código.

```
par(mfrow = c(1, 2))
library(Epi)
ROC(
  form = diag ~ edad + dep + sexo + tdolor, data = cleveland,
  plot = "sp"
)
ROC(
  form = diag ~ edad + dep + sexo + tdolor, data = cleveland,
  plot = "ROC", las = 1
)
```

La Figura 16.2 (izquierda) muestra que tomando como punto de corte una probabilidad cercana a 0.45 se obtienen valores de sensibilidad y especificidad alrededor de 0.8. No obstante, se deben evaluar los costes y riesgos de una mala clasificación (por ejemplo, dar tratamiento cuando no hace falta, y no darlo cuando hace falta).

La Figura 16.2 (derecha) representa la curva ROC de `reg.log2`. La curva está por encima de la diagonal, con lo que es mejor que un clasificador aleatorio. El AUC es de 0.878, con una

Figura 16.2: Gráfico de sensibilidad y especificidad según puntos de corte de discriminación (izquierda) y curva ROC (derecha) para el segundo modelo de regresión logística

sensibilidad de 78.4 % y una especificidad de 84.1 %. Si se realiza el análisis para `reg.log` se comprobará que el AUC es de 0.759, por lo que el segundo modelo es mejor para discriminar. Además, el gráfico indica el valor del 'punto de corte óptimo', 0.518, el que proporciona el mayor AUC, el citado 0.878, con sus correspondientes valores de sensibilidad, especificidad, etc. También presenta las estimaciones y errores estándar (*s.e.*) de los coeficientes del modelo considerado.

16.5.2. Ejemplo de regresión de Poisson

A partir del mismo conjunto de datos, `cleveland`, se considera como variable a explicar, variable respuesta, `dhosp`, el número de días de hospitalización de un paciente. Y como variables explicativas: `diag`, `edad`, `sexo` y `tdolor`, que son de distinto tipo, lo que permitirá ilustrar sus distintas interpretaciones.

Visualización ilustrativa

```
p1 <- ggplot(cleveland, aes(dhosp, fill = diag)) +
  geom_bar(position = "dodge")
p2 <- ggplot(cleveland, aes(x = factor(dhosp), y = edad)) +
  geom_boxplot()
library(patchwork)
p1 + p2
```

Figura 16.3: Gráfico de barras de 'dhosp' separado por 'diag' (izquierda) y gráficos de cajas de 'edad' separados por 'dhosp' (derecha).

A la vista de los gráficos de la Figura 16.3, generados con las sentencias anteriores, `diag` parece ser una buena variable predictora del número de días de hospitalización, mientras que `edad` no tiene una influencia tan clara.

Ajuste e interpretación

```
reg.pois <- glm(dhosp ~ diag + edad + sexo + tdolor,
  data = cleveland, family = "poisson"
)
```

Al especificar `family = "poisson"` la función `glm` selecciona automáticamente la función de enlace canónica apropiada: el logaritmo (efectos multiplicativos).

```
summary(reg.pois)$coef
#>                  Estimate  Std. Error     z value     Pr(>|z|)
#> (Intercept) -0.2230728726 0.332125111 -0.67165314 5.018045e-01
#> diag1        1.0902902179 0.109731744  9.93595999 2.903654e-23
#> edad         0.0001026277 0.004877904  0.02103931 9.832143e-01
#> sexo1        0.2160286255 0.103297627  2.09132226 3.649919e-02
#> tdolor2      0.1408464230 0.205232539  0.68627725 4.925383e-01
#> tdolor3      0.1217757202 0.189404790  0.64293897 5.202637e-01
#> tdolor4      0.1215388103 0.178729482  0.68001546 4.964947e-01
exp(coef(reg.pois))[-1]
#>    diag1     edad    sexo1  tdolor2  tdolor3  tdolor4
#> 2.975137 1.000103 1.241138 1.151248 1.129501 1.129233
```

De todas las variables introducidas en el modelo sólo `diag` y `sexo` son significativas (al 5 %). Se confirma así lo visto en la Figura 16.3 (derecha), que `edad` no influye en la respuesta. Al ser ambas variables significativas de tipo dicotómico su interpretación, como ya ocurría en regresión logística, se hace en función del 'nivel/grupo de referencia', y *ceteris paribus* (en todos los casos).

- El coeficiente de `diag` es 1.0903 pero se le debe aplicar la exponencial para tener como unidades de medida días. Así, el número medio de días de estancia en el hospital es 2.98 veces mayor con `diag=1` que con `diag=0` (grupo tomado como referencia).
- Similar para `sexo`, un hombre (`sexo=1`) se espera que esté, en media, 1.24 días por cada día que esté una mujer.
- De la misma manera se interpretarían los coeficientes asociados a `tdolor`, cada uno de ellos expresaría la diferencia con los pacientes del grupo `tdolor=1`.
- Aunque `edad` no es significativa, y la magnitud es ínfima, se da su interpretación, al ser la única variable continua: *ceteris paribus* por cada año que aumente la `edad` el número medio de días en el hospital se ve multiplicado por 1.0001 (ínfimo)[17].

Para posteriores comparaciones se considera otro modelo en el que se elimina `tdolor` (por no ser significativa):

[17] aunque el incremento fuese de 20 años (como se hizo en el ejemplo de regresión logística), por cada día que esté hospitalizado el más joven, el de 20 años más estará $e^{\beta_2 * 20} = 1.0021$ días.

```
reg.pois2 <- glm(dhosp ~ diag + sexo + edad,
  data = cleveland, family = "poisson"
)
```

Adecuación

En las salidas obtenidas hay información sobre la *Null* y la *Residual deviance*, con las que se puede realizar un contraste de comparación de modelos (el simple frente al elaborado).

```
pchisq(reg.pois$deviance, reg.pois$df.residual, lower.tail = F)
#> [1] 0.1325654
pchisq(reg.pois2$deviance, reg.pois2$df.residual, lower.tail = F)
#> [1] 0.155157
```

Al ser los p-valores superiores a 0.05 ambos modelos se pueden considerar que explican 'mejor' que el modelo nulo.

Predicción

Con los modelos ajustados, y comprobada su adecuación, se pueden predecir valores, que en este caso serán el número *medio* de días de hospitalización.

```
pacientes <- data.frame(
  diag = c("1", "0", "1", "0"),
  edad = c(50, 50, 50, 50),
  sexo = c("1", "1", "0", "0"),
  tdolor = c("4", "4", "4", "4")
)
predict(reg.pois, pacientes, type = "response")
#>         1         2         3         4
#> 3.3532035 1.1270752 2.7017171 0.9080983
```

Se han escogido arbitrariamente valores de las variables explicativas para 4 `pacientes`: 2 de ellos con accidente coronario (`diag=1`), un hombre (`sexo=1`) y una mujer y otros 2 sin accidente, también un hombre y una mujer, los cuatro con 50 años y asintomáticos (`tdolor=4`). La predicción obtenida indica que el hombre con accidente coronario estará hospitalizado más días, en media, que el resto, aunque le sigue de cerca la mujer con accidente coronario.

También se pueden dibujar las predicciones de todo el conjunto de datos, que `glm` ha guardado como `fitted.values`. Para el modelo `reg.pois` sería:

```
cleveland$hat <- reg.pois$fitted.values
ggplot(cleveland, aes(x = edad, y = hat, colour = diag)) +
  geom_point() +
  labs(x = "Edad", y = "Días hospitalización")
```

Figura 16.4: Predicciones de todo el conjunto de datos del modelo reg.pois (izquierda) y reg.pois2 (derecha)

La Figura 16.4 muestra las predicciones de los 303 individuos, para cada uno de los dos modelos de regresión de Poisson considerados. Cabe recordar que en el segundo se ha eliminado la variable `tdolor`. A la vista de dichos gráficos se puede concluir que es `tdolor` la que genera las distintas 'lineas horizontales' de puntos en torno a `hosp=3` por un lado y en torno a `hosp=1` por otro lado.

Resumen

En este Capítulo se ha introducido el modelo de regresión lineal *generalizado*, indicado cuando las respuestas no son gaussianas (normales). En particular:

- Se ha introducido la función de enlace, que juega un papel importante en estos modelos.
- Se describen los casos particulares de regresión logística y de regresión de Poisson.
- Se muestra el uso de `R` para el ajuste de estos modelos.
- Se ilustra la interpretación de los coeficientes, tanto asociados a variables continuas como a categóricas, y los demás resultados obtenidos con `R` mediante casos prácticos.
- Se incluye el uso de la regresión logística como clasificador y se dan indicaciones de su análisis mediante la curva ROC y el AUC.

Capítulo 17

Modelos aditivos generalizados

María Durbán y Víctor Casero

17.1. Introducción

Los modelos lineales, o los lineales generalizados (GLM), vistos en capítulos anteriores son deseables porque son simples de ajustar, se entienden fácilmente, y se dispone de técnicas para contrastar las hipótesis del modelo. Sin embargo, cuando los datos no están relacionados de forma lineal con las variables explicativas no tiene sentido utilizar los modelos de regresión lineal, se necesitan modelos que flexibilicen esta relación, que se puede expresar así:

$$Y = \beta_0 + f(X) + \varepsilon$$

Puede que la función $f()$ sea conocida de antemano, como ocurre en muchos modelos biológicos, donde existe una dependencia de tipo exponencial, $f(x) = e^{\beta_0+\beta_1 x}$. En otras ocasiones dicha función es desconocida y se puede utilizar una aproximación. Por ejemplo, la muy utilizada regresión polinómica:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_p X^p + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{17.1}$$

Sin embargo, la regresión polinómica tiene un gran inconveniente, que no se realiza a nivel local: cada vez que se cambia un coeficiente del modelo, el cambio impacta a los valores ajustados en todo el rango de la variable explicativa.

La alternativa es utilizar *suavizadores* de las variables explicativas. Se define así el **modelo aditivo** (en el caso de variable respuesta Gaussiana) para al menos dos variables explicativas:

$$Y = f(X_1) + \ldots + f(X_j) + \epsilon.$$

Las funciones f incluyen también a las funciones lineales vistas en el Capítulo 16. Los modelos aditivos generalizados, GAM, extienden lo anterior a respuestas no gaussianas, como lo hacen los GLMs respecto de los modelos lineales con respuesta gaussiana (véase el Capítulo 17).

17.2. Splines con penalizaciones

Las técnicas de suavizado tienen como objetivo extraer las tendencias (o señales) subyacentes en los datos. Un suavizador o *smoother* es una herramienta que representa la tendencia de la variable respuesta Y como función de una o más variables predictoras. Tiene menos variabilidad que Y, por eso se le llama suavizador (la regresión lineal es una suavizador llevado al extremo). La curva en un punto dado sólo depende de las observaciones en ese punto y de las observaciones vecinas. Estás son algunas de las técnicas de regresión no-paramétrica existentes:

1. Regresión polinomial local con pesos, *lowess*.
2. Kernels.
3. Splines.

Este libro se centra en el caso de los Splines que son los suavizadores más utilizados y en particular en los splines con penalizaciones (*P-splines*). Estos se basan en representar la función f mediante una base de funciones y añadir una penalización a la hora de estimar el modelo de manera que se pueda controlar la variabilidad de la curva que se quiere estimar. Hay muchas maneras de representar una función a través de una base, una de las mejores opciones es el uso de los B-splines (De Boor, 2001) por sus buenas propiedades numéricas. La penalización se construye a partir de la derivada de la curva que se quiere penalizar, en general, se utilizan penalizaciones de orden 2 lo que implica que se está penalizando todo aquello que no es lineal en la función, es decir, si la penalización es muy grande la curva estimada sería simplemente una línea recta. La penalización está controlada por un parámetros llamado **parámetro de suavizado**.

A la hora de ajustar este tipo de modelos hay dos elecciones importantes:

- Tamaño de la base: la mayoría de los autores utilizan como regla:

$$\text{número de nodos} = min\{40, \text{valores únicos de } x/4\}$$

  (por ejemplo si se tienen 100 datos, se elegirían 100/4=25 nodos)
- Parámetro de suavizado: Se elegirá utilizando máxima verosimilitud junto con el resto de parámetros del modelo. La Figura 17.1 muestra el impacto que el parámetro de suavizado tiene en el ajuste final de la curva (los datos corresponden al dataset `fossil` del paquete `Semipar`).

17.3. Aspectos metodológicos

Al igual que en el caso de los modelos lineales y los modelos GLM, en los modelos GAM es necesario conocer algunos aspectos metodológicos que son fundamentales llevar a cabo un ajuste correcto de los modelos, y entender los resultados obtenidos en el ajuste. A continuación se muestran los más relevantes

Figura 17.1: Regresión con P-splines para diferentes valores del parámetro de suavizado

17.3.1. Estimación de los paraámetros del modelo

La estimación de los modelos GAM se lleva a cabo mediante máxima verosimilitud penalizada. Supóngase el caso de una sola variable explicativa y se quiere ajustar el modelo:

$$Y = f(X) + \epsilon.$$

Como se comentó anteriormente, los modelos GAM tiene como punto de partida la representación básica de funciones, es decir, se busca transformar el modelo de tal forma que $f(X)$ sea el producto de una matriz multiplicada por unos coeficientes, es decir, se elige una *base* (una matriz $\mathbf{B}$) que permita escribir la función f como una combinación lineal de sus elementos (los elementos de esta base son conocidos)

$$f(X) = \sum_{j=1}^{k} b_j(X)\beta_j,$$

o en forma matricial:

$$f(X) = \mathbf{B}\boldsymbol{\beta}.$$

y los parámetros $\boldsymbol{\beta}$ se estiman minimizando la siguiente expresión:

$$(\boldsymbol{y} - \boldsymbol{B\theta})'(\boldsymbol{y} - \boldsymbol{B\theta}) + \lambda \boldsymbol{P},$$

Donde $\boldsymbol{P}$ es la matriz de penalización y λ es el parámetro de suavizado. Dado un parámetro de suavizado, los parámetros estimados vienen dados por:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{B}^T\boldsymbol{B} + \lambda\boldsymbol{P})^{-1}\boldsymbol{B}'\boldsymbol{y}$$

y los valores ajustados se obtendrían como: $\hat{\boldsymbol{y}} = \underbrace{\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{B} + \lambda\boldsymbol{P})^{-1}\boldsymbol{B}'}_{\boldsymbol{H}}\boldsymbol{y}$. La matriz $\boldsymbol{H}$ juega un papel importante ya que la suma de su diagonal da una idea de cómo de compleja es la curva

ajustada, es lo que se llaman los *grados de libertad efectivos* (los cuales no se corresponden al número de parámetros ajustados).

17.3.2. Inferencia sobre las funciones suaves

Si nuestro interés es saber si existe una relación estadísticamente significativa entre la variable explicativa X_j y la variable respuesta Y:

$$\begin{aligned} H_0 &: f_j = 0 \quad \text{(no efecto)} \\ H_1 &: f_j \neq 0 \quad \text{(efecto)} \end{aligned}$$

- Dado que las funciones f_j dependen de los coeficientes que acompañan a las bases de B-splines, el contraste anterior es equivalente a:

  $$\begin{aligned} H_0 &: \theta_j = 0 \qquad \forall j \\ H_1 &: \theta_j \neq 0 \quad \text{para al menos un } j \end{aligned}$$

  La distribución del estadístico de contraste dependerá de si la variable respuesta sigue una distribución Normal o no:

- Si los datos son Normales, el estadístico de contraste sigue un distribución F.
- En otro caso sigue una distribución χ^2.

Comparación de modelos

1. Modelos Anidados. La comparación se basa, al igual que en los GLM, en la diferencia en la *deviance* residual. Si se quieren comparar dos modelos m_1 y m_2 (donde $m_1 \subset m_2$), entonces:

   - En el caso de datos Normales:

     $$\frac{(DR(m_1) - DR(m_2))/(df_2 - df_1)}{DR(m_2)/(n - df_2)} \approx F_{(df_2 - df_1),(n - df_2)}$$

     donde DR es la deviance residual (Suma de cuadrados residual) y df son los grados de libertad asociados con cada modelo.

   - En otro caso:

     $$DR^*(m_1) - DR^*(m_2) \approx \chi^2_{df_2 - df_1}$$

     donde DR^* es la deviance residual escalada.

2. Modelos no anidados

   En este caso los contrastes anteriores no son válidos y se utilizarán criterios basados en el AIC (Criterio de Información de Akaike).

17.3.3. Suavizado mutidimensional y para datos no Gaussianos

Para el suavizado en 2 dimensiones (o más) también se necesita una base y una penalización. El modelo sería:

$$\boldsymbol{Y} = f\left(\boldsymbol{X}_1, \boldsymbol{X}_2\right) + \epsilon_i$$

donde $f()$ es una función suave de las dos covariables $\boldsymbol{X}_1$ y $\boldsymbol{X}_2$. Dicha función se aproxima mediante el producto tensorial de las bases de B-splines marginales para cada una de las covariables. los términos de suavizado multidimensional se pueden combinar con términos unidimensionales y términos lineales.

La extensión de los modelos de suavizado al caso en el que la variable respuesta es no Gaussiana, se hace de forma similar al caso lineal, cuando se pasa de un modelo de regresión lineal a un GLM.

- Como en el caso de los GLMs $g(\boldsymbol{\mu}) = \boldsymbol{\eta} = f(\boldsymbol{X}) = \boldsymbol{B}\boldsymbol{\beta}$
- Se añade la penalización a la función de verosimilitud

$$\ell_p(\beta) = \ell(\beta) + \lambda \beta^T P \beta,$$

  donde $\ell(\beta)$ es la log-verosimilitud.

17.4. La función `gam` del paquete `mgcv`

Aunque hay muchas librerías disponibles la principal es `mgcv` que implementa una gran variedad de modelos de suavizado a través de la función `gam` (generalized additive models). La principal referencia para esta sección es el libro de Wood (2006).

```
gam(formula, method = "", select = "", family = gaussian())
```

- `formula` es el argumento principal de esta función, es la ecuación del modelo: por ejemplo, `y ~ x1+x2+s(x3)`.
  - Lo primero que se tiene que elegir es la base a utilizar para representar las funciones suaves, `s(x)` (ver `?s` o `?smooth.terms`), o `te(x1,x2)` en el caso de suavizado multidimensional. Por defecto son los llamados thin plate splines.
    El tipo de base usada se puede modificar utilizando el argumento `bs` dentro de `s(x, bs = "ps")`, en este caso `ps` indica el uso de B-splines con penalizaciones. En la siguiente tabla se describen otras alternativas:

`bs`	**Descripción**
`"tp"`	Thin Plate Regression Splines
`"ts"`	Thin Plate Regression Splines con regularización

bs	Descripción
"\cr"	Spline cúbicos de regresión
" crs"	Spline cúbicos de regresión con regularización
" cc"	Spline cíclicos
"ps"	P-splines

- `m` orden de la penalización, por defecto es 2.
- `k` número de nodos para construir la base. El número por defecto suele ser demasiado bajo por lo que siempre se recomienda que el usuario elija el número.
- `by` una variable numérica o factor de la misma dimensión de cada covariable que permite hacer interacciones entre curvas y variables
- `id` se utiliza para forzar que diferentes términos suaves utilicen la misma base y la misma cantidad de suavizado
- `method` método para estimar el parámetro de suavizado, se puede elegir entre: `REML`, `ML`, `GCV.Cp`, `GACV.Cp` (en la práctica se prefiere "REML")
- `family` permite elegir la distribución de los datos (binomial, Poisson, etc.), por defecto asume datos Gaussianos.
- `select=TRUE` : Contrasta si la variable debe entrar o no en el modelo

17.5. Casos prácticos

En este apartado se verán una serie de aplicaciones que permiten mostrar los diferentes usos de este tipo de modelos.

17.5.1. Modelo unidimensional con `fossil`

Se empezará ilustrando el uso de la función `gam` con el conjunto de datos `fossil` del paquete `SemiPar`. El objetivo es estimar la relación entre la edad de los fósiles y la proporción de isotopos de estroncio.

A la vista de la Figura 17.2, es claro que se necesita ajustar una curva (y no una línea) para estimar la relación entre ambas variables. Para ello se utiliza la función `gam` que devuelve un objeto de tipo `"gam"`, que se puede usar con las típicas funciones `print`, `summary`, `fitted`, `plot`, `residuals`, etc...

```
library(mgcv)
fit.gam <- gam(Y ~ s(X, k = 25, bs = "ps"), method = "REML", select = TRUE)
summary(fit.gam)
#>
#> Family: gaussian
#> Link function: identity
#>
```

Figura 17.2: Edad de fósiles con respecto a la proporción de isótopos de estroncio

```
#> Formula:
#> Y ~ s(X, k = 25, bs = "ps")
#>
#> Parametric coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 7.074e+03  2.435e-02  290504   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>        edf Ref.df     F p-value
#> s(X) 10.22     24 35.89  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.891   Deviance explained = 90.2%
#> -REML = 23.946  Scale est. = 0.062849  n = 106
```

Como se puede ver se han especificado la variable respuesta (Y, proporción de estroncio) y la variable explicativa (X, edad) mediante un *spline*, `s()`, de tipo penalizado, `ps`, con 25 nodos, indicando `REML` como método para estimar el parámetro de suavizado.

De las 2 partes de la salida anterior, en la primera aparecen los términos que entran como lineales en el modelo (en este caso solo la ordenada en el origen) y en la parte de abajo aparecen los términos de suavizado. Como se indicó anteriormente, dado que se ha usado `select=TRUE`, se está contrastando si la variable `edad` debe entrar en el modelo o no. En este caso es claro que ha de entrar ya que el p-valor de `s(x)` pequeño y los grados de libertad asociados son aproximadamente 10, lo que indica que está lejos de la linealidad.

La función `gam.check` devuelve los gráficos de residuos usuales (residuos frente a valores ajustados, gráficos de cuantiles para comprobar la normalidad, etc.), pero además da información sobre el proceso de ajuste del modelo.

```
gam.check(fit.gam)
```

Figura 17.3: Gráficos de residuos obtenidos con 'gam.check'

```
#>
#> Method: REML   Optimizer: outer newton
#> full convergence after 5 iterations.
#> Gradient range [-4.557319e-06,5.900236e-06]
#> (score 23.94602 & scale 0.06284944).
#> Hessian positive definite, eigenvalue range [4.557347e-06,53.03185].
#> Model rank =  25 / 25
#>
#> Basis dimension (k) checking results. Low p-value (k-index<1) may
#> indicate that k is too low, especially if edf is close to k'.
#>
#>        k'  edf k-index p-value
#> s(X) 24.0 10.2    1.03    0.59
```

El test que aparece en la parte de abajo está contrastando si el número de nodos elegido es suficiente. Si el valor de `k` está muy próximo a `edf` entonces se debería reajustar el modelo con más nodos.

Si se quiere dibujar la función suave se utiliza el comando `plot()`.

```
plot(fit.gam, shade = TRUE, seWithMean = TRUE, pch = 19, 1, cex = .55)
```

Figura 17.4: Curva ajustada e intervalo de confianza

17.5.2. Modelo aditivo con `airquality`

En esta sección se analizan de nuevo los datos `airquality` (ver ?airquality[1]) que consisten en 154 medidas de calidad del aire en Nueva York, de Mayo a Septiembre 1973. El objetivo es establecer la relación entre las variables meteorológicas y la cantidad de ozono en la atmósfera. Ya se ha analizado la relación en el Capítulo 16 de Modelos lineales, donde los ajustes lineales realizados eran satisfactorios, pero se encontraban problemas en los residuos del modelo, que no daban validez a dicha modelización. Allí se sugería que la relación entre la variable respuesta y alguna explicativa fuese no lineal. Además se consideró la transformación logarítmica de la variable `Ozone`, que adolece de simetría, y con dicha trasformación se obtenía una distribución más similar a la distribución Normal.

Por lo tanto se va a ajustar el modelo incluyendo las variables explicativas de forma suave, en particular se van a incluir las variables `Wind`, `Temp` y `Solar.R`. Las variable `Wind` y `Temp` tienen sólo 31 y 40 valores únicos respectivamente, aunque el dataset tiene 154 valores, por eso se decide utilizar 10 nodos y no más, mientras que para la variable `Solar.R` se utilizan 20.

```
airq.gam <- gam(
  log(Ozone) ~ s(Wind, bs = "ps", k = 10) +
    s(Temp, bs = "ps", k = 10) +
    s(Solar.R, bs = "ps", k = 20),
  method = "REML", select = TRUE, data = airquality, na.action = na.omit
```

[1]conjunto de datos incluido con la instalación base de R.

```
)
summary(airq.gam)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log(Ozone) ~ s(Wind, bs = "ps", k = 10) + s(Temp, bs = "ps",
#>     k = 10) + s(Solar.R, bs = "ps", k = 20)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  3.41593    0.04586   74.49   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>               edf Ref.df     F  p-value
#> s(Wind)     2.318      9 2.255 3.13e-05 ***
#> s(Temp)     1.852      9 6.128  < 2e-16 ***
#> s(Solar.R) 2.145     19 1.397 2.31e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.689   Deviance explained = 70.7%
#> -REML = 86.106  Scale est. = 0.23342   n = 111
```

Los resultados indican que todas las variables son significativas (p-valores pequeños), estando la variable `Temp` próxima a la linealidad. El R^2 ajustado es 0.68 por lo que el modelo ajusta moderadamente bien los datos. La Figura 17.5 muestra la tres curvas ajustadas, incluyendo los llamados *residuos parciales* que corresponden a, por ejemplo en el caso del gráfico del viento, $log(Ozone) - \hat{\beta}_0 - \hat{f}(Temp) - \hat{f}(Solar.R)$, es decir, lo que queda sin explicar después de haber ajustado los demás términos del modelo.

```
par(mfrow = c(2, 2))
plot(airq.gam, residuals = TRUE, scheme = 1)
```

17.5.3. Modelo semiparamétrico con `onions`

Es un caso particular del modelo aditivo anterior pues en este tipo de modelos todas las variables entran de forma lineal excepto una.

$$\boldsymbol{Y} = \beta_0 + \beta_1 \boldsymbol{X}_1 + \ldots + \beta_{j-1} \boldsymbol{X}_{j-1} + f(\boldsymbol{X}_j) + \epsilon$$

Figura 17.5: curvas estimadas para 'Wind', 'Temp' y 'Solar'

La forma de ajustar el modelo es exactamente igual a la anterior. Pero hay un caso que merece especial interés, cuando en la parte paramétrica se incluye una variable categórica con dos o más niveles. Al igual que en el caso de regresión lineal se puede plantear si se quieren ajustar dos rectas paralelas (modelo aditivo) o no paralelas (modelo con interacción).

Para ilustrar este caso se acude al data.frame `onions` (librería `SemiPar`). Contiene 84 observaciones de un experimento sobre la producción de un tipo de cebolla en dos localidades (Purnong Landing y Virginia). El objetivo es relacionar el logaritmo de la producción de cebollas con la densidad de plantas por metro cuadrado. El modelo lineal básico sería:

$$\log(\text{yield}_i) = \beta_0 + \beta_1 \text{location}_{ij} + \beta_2 \text{dens}_i + \epsilon_i$$

donde

$$\text{location}_{ij} = \begin{cases} 0 & \text{si la observación } i \text{ es de Purnong Landing} \\ 1 & \text{si la observación } i \text{ es de Virginia} \end{cases}$$

Se comienza por ajustar el siguiente modelo:

$$\log(\text{yield}_i) = \beta_0 + \beta_1 \text{location}_{ij} + f(\text{dens}_i) + \epsilon_i$$

```
library(mgcv)
library(SemiPar)
data(onions)
# Se indica a R que la variable location es categórica
onions$location <- factor(onions$location)
# Se recodifica la variable
levels(onions$location) <- c("Purnong Landing", "Virginia")
fit1 <- gam(log(yield) ~ location + s(dens, k = 20, bs = "ps"),
  method = "REML", select = TRUE, data = onions
```

```
)
summary(fit1)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log(yield) ~ location + s(dens, k = 20, bs = "ps")
#>
#> Parametric coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)       4.85011    0.01688  287.39   <2e-16 ***
#> locationVirginia -0.33284    0.02409  -13.82   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>           edf Ref.df     F p-value
#> s(dens) 4.568     19 72.76  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.946   Deviance explained = 94.9%
#> -REML = -54.242  Scale est. = 0.011737  n = 84
```

En este caso se ve que en la parte lineal aparecen dos parámetros, ambos significativos, la ordenada en el origen y el parámetro que corresponde a la variable categórica `location`, concretamente el parámetro correspondiente a la categoría Virginia que es negativo indicando que la producción media para la otra localidad es mayor que la producción media en Virginia. El término de suavizado es significativo también.

En este caso si se utiliza la función `plot.gam`, sólo dibujará una curva, pues las curvas para las dos localizaciones son paralelas y la diferencia entre ellas es igual al valor del parámetro de la variable localización. Para dibujar las curvas para cada localización se utiliza la función `plot_smooth` de la librería `tidymv`, los argumentos son, primero el modelo, después la variable con respecto a la que se está ajustando la curva y por último la variable categórica.

```
library(tidymv)
plot_smooths(fit1, dens, location)
```

Asumir curvas paralelas para ambas localidades implica que el descenso en producción de cebollas a medida que aumenta la densidad de plantas es el mismo para ambas localidades, y esto no tienen por qué ser cierto, para relajar esta hipótesis se puede ajustar un modelo con interacción (de manera similar a lo que se hace en el caso de regresión lineal):

$$\log(\text{yield}_i) = \beta_0 + \beta_1 \text{location}_{ij} + f(\text{dens}_i)_{L(j)} + \epsilon_i$$

Figura 17.6: curvas ajustadas para ambas localidades

donde

$$L(j) = \begin{cases} 0 & \text{si la } i\text{-ésima observación es de Purnong Landing} \\ 1 & \text{si la } i\text{-ésima observación es de Virginia} \end{cases}$$

Para hacerlo en `R`, se introduce el argumento `by=location` dentro de la curva

```
fit2 <- gam(log(yield) ~ location + s(dens, k = 20, bs = "ps", by = location),
  method = "REML", data = onions
)
summary(fit2)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log(yield) ~ location + s(dens, k = 20, bs = "ps", by = location)
#>
#> Parametric coefficients:
#>                   Estimate Std. Error t value Pr(>|t|)
#> (Intercept)        4.84415    0.01603  302.19   <2e-16 ***
#> locationVirginia -0.33018    0.02270  -14.54   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                                   edf Ref.df     F p-value
#> s(dens):locationPurnong Landing 3.096  3.834 176.9  <2e-16 ***
#> s(dens):locationVirginia        4.742  5.795 153.0  <2e-16 ***
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.952   Deviance explained = 95.7%
#> -REML = -58.541  Scale est. = 0.010446  n = 84
```

Ahora aparecen dos términos suaves, uno para cada localidad, de modo que estas curvas no tienen por qué ser paralelas, sino que cada una se ajustará a la forma que tengan los datos. En este caso la Figura 17.7, generada de nuevo con `plot_smooths`, muestra como las curvas se van alejando a medida que aumenta la densidad de plantas.

Figura 17.7: curvas ajustadas para ambas localidades permitiendo que no sean paralelas

Para finalizar se comparan ambos modelos con el criterio AIC.

```
AIC(fit1)
#> [1] -125.2307
AIC(fit2)
#> [1] -131.2181
```

Dado que el menor valor se alcanza en el segundo modelo, se escogería el modelo que incluye la interacción entre la variable densidad y la localidad.

17.5.4. Modelo aditivo generalizado y multidimensional, con `smacker`

Se van a analizar los datos `smacker` del paquete `sm`. El objetivo es ver cómo influyen las condiciones del mar (temperatura de agua, etc.) en la ausencia o presencia de huevos de jurel en el mar cantábrico, además, se utilizará la posición geográfica (latitud y longitud) como covariables para captar el efecto espacial.

Figura 17.8: Localización donde se midió la ausencia/presecia de huevos de jurel

Dado que la variable respuesta es dicotómica se va a utilizar un modelo de regresión logística en el que se va a flexibilizar la linealidad de las variables y además se usará una superficie para estimar el efecto de la localización como una función en dos dimensiones (en función de la posición geográfica: latitud y longitud). En este caso en vez de usar el término `te()` se usa `s()` también para el caso de 2 dimensiones. La diferencia fundamental con `te()` es que `s()` asume un suavizado isotrópico, es decir el mismo parámetro de suavizado para la Latitud y Longitud. No se debe usar `s()` para el suavizado en dos dimensiones si las covariables están medidas en unidades diferentes (en este caso como sí lo están, se puede usar el suavizado isotrópico)

```
logit1 <- gam(
  Presence ~ s(smack.long, smack.lat, k = 60) +
    s(ldepth) + s(Temperature),
  family = binomial, select = TRUE, data = smacker
)
par(pty = "s", mfrow = c(1, 2))
plot(logit1, scheme = 2)
```

La relación entre la temperatura y la probabilidad de presencia de huevos es no lineal mientras que es lineal para la profundidad. El R^2 es 0,4 por lo que sería necesario recoger más variables para poder obtener buenas predicciones.

Si se quisieran obtener las probabilidades predichas se haría con la función `predict`.

```
prob <- predict(logit1, type = "response")
```

Resumen

En este capítulo se han introducido los modelos aditivos generalizados, en particular:

- Se han presentado distintos aspectos metodológicos relacionados con la estimación e inferencia en modelos GAM.
- Se ha mostrado el uso de `R` para el ajuste de este tipo de modelos.
- Se han presentado diversos casos prácticos que ilustran la versatilidad de estos modelos para analizar datos complejos.

Capítulo 18

Modelos mixtos

María Durbán y Víctor Casero

18.1. Conceptos básicos

Los modelos mixtos (MMs) para variables de respuesta continuas son modelos estadísticos en los que los residuos siguen una distribución Normal pero puede que no sean independientes o no tengan varianza constante. Son necesarios en muchas situaciones, sobre todo en experimentos donde se realiza algún tipo de muestreo:

1. Estudios con datos agrupados, como por ejemplo, alumnos en una clase, individuos en una ciudad.
2. Estudios longitudinales o de medidas repetidas, donde un individuo es medido repetidamente a lo largo del tiempo o bajo condiciones distintas.

Este tipo de estudios se pueden encontrar en diferentes áreas como la Medicina, Biología, Ciencias Experimentales y Sociales.

18.1.1. Tipo y estructura de los datos

La estructura de los datos con la que se trabaja es el factor determinante para saber si se han de utilizar modelos mixtos, y en su caso, qué tipo de modelo.

Datos jerárquicos (o agrupados)

En este tipo de datos la variable dependiente (de respuesta, de interés) se mide una sola vez en cada individuo (la unidad de análisis), y los individuos está agrupados (o anidados) en unidades mayores. Muchos tipos de datos tienen una estructura jerárquica: alumnos en escuelas, personas en municipios, pacientes en hospitales, plantas en una parcela...

Las jerarquías son una forma de representar la relación de dependencia que hay entre los individuos y los grupos a los que pertenecen. Por ejemplo, supóngase que se quiere hacer un estudio sobre el tiempo de recuperación en pacientes hospitalizados por COVID-19 en diferentes hospitales. Se tiene la siguiente estructura con dos niveles:

- Muchos individuos en el nivel 1 (pacientes)
- Agrupados en unas pocas unidades en el nivel 2 (hospitales)

Las estructuras multinivel pueden aparecer también como consecuencia del diseño del estudio que se está llevando a cabo. Por ejemplo, una encuesta sobre el estado de salud puede dar lugar a un diseño a tres niveles: primero se muestrean regiones, luego distritos y después individuos.

En cada nivel de la jerarquía se pueden medir variables. Algunas estarán medidas en su nivel *natural*, por ejemplo, en el nivel del hospital se podría medir el tamaño, y al nivel de los pacientes se podría medir su situación socio-económica. Además, se pueden mover las variables de un nivel a otro mediante agregación o desagregación:

- **Agregación**: La variable al nivel más bajo se mueve a un nivel más alto, por ejemplo, se pueden asociar a cada hospital la media del nivel socioeconómico de sus pacientes.
- **Desagregación**: Mover las variables a un nivel más bajo, por ejemplo, asignarle a cada paciente una variable que indique el tamaño de su hospital de referencia.

Medidas repetidas y datos longitudinales

En este tipo de datos la variable dependiente se mide más de una vez a un mismo individuo (Singer and Willet, 2003). Por ejemplo, se miden los niveles de glucosa de un enfermo antes y después de haberle inyectado insulina. Este tipo de datos también puede ser considerados como datos multinivel (o jerárquicos) donde el Nivel 2 representa a los individuos y el Nivel 1 representa a las diferentes medidas tomadas. Dado que las medidas se toman a un mismo individuo, es probable que dichas medidas no sean independientes, por lo que utilizar un modelo lineal ordinario no sería apropiado.

Por **datos longitudinales**, se entienden datos en los que la variable dependiente se ha medido en distintos instantes de tiempo en cada una de las unidades de análisis. En algunos casos, cuando la variable dependiente se mide a lo largo del tiempo, puede ser difícil identificar si los datos son medidas repetidas o datos longitudinales. Desde el punto de vista del análisis de los datos mediante MMs esta distinción no es un elemento crítico. Lo importante es que en ambos tipos de datos la variable dependiente se ha medido repetidas veces en la misma unidad de análisis, y que por tanto las observaciones estarán correlacionadas.

18.1.2. ¿Efectos fijos o aleatorios?

En un modelo mixto la clave se encuentra en la distinción entre efectos fijos y aleatorios (Snijers, 2003). Esto es importante porque la inferencia y el análisis de ambos efectos es distinta.

Los **efectos fijos** son variables en las cuales el investigador ha incluido sólo los niveles (o tratamientos) que son de su interés. Por ejemplo, en un experimento se puede estar interesados en comparar dos grupos, uno al que se le aplica un tratamiento y otro de control. En este caso, el objetivo del estudio compara los grupos y no interesa generalizar los resultados a otros tratamientos que podrían haber sido incluidos. Otro ejemplo sería el caso en el que se hace un encuesta y se eligen 10 ciudades. Si sólo interesan los resultados para esas 10 ciudades y no se quieren generalizar los resultados al resto de ciudades que podrían haber sido seleccionadas, la variable 'ciudad' será un efecto fijo. Si se eligen las ciudades de forma aleatoria de una población grande de ciudades se consideraría la variable 'ciudad' como un **efecto aleatorio**.

Una cantidad se considera aleatoria cuando cambia sobre las unidades de una población. Cuando un efecto en un modelo estadístico es considerado aleatorio, se está asumiendo que se quieren extraer conclusiones sobre la población de la cual se han elegido las unidades observadas, y no se tiene interés en esas unidades en particular. En este contexto se habla de *intercambiabilidad*, en el sentido de que se podría cambiar una unidad de la muestra por otra de la población y sería indiferente. Este es el caso de los factores de agrupamiento o diseño, como son los bloques en un experimento agrícola, o los días cuando un experimento se lleva a cabo en días distintos, o un técnico de laboratorio cuando hay varios haciendo el experimento; también lo serían los sujetos en un diseño de medidas repetidas o las localizaciones donde se recogen muestras en un río, si el objetivo es generalizar a todo el río.

Los métodos estándar utilizados para construir tests e intervalos de confianza para los efectos fijos, no son válidos para los efectos aleatorios, pues los efectos observados son sólo una muestra de todos los posibles efectos.

La clave para distinguir, estadísticamente hablando, entre efectos fijos y aleatorios es si los niveles de la variable se pueden interpretar como extraídos de una población con una cierta distribución de probabilidad. En el caso de un efecto fijo, normalmente interesará comparar los resultados de la variable dependiente para los distintos niveles de la variable explicativa, es decir, interesará la diferencia entre las medias. En el caso de efectos aleatorios, no interesa específicamente comparar si las medias son distintas, sino cómo el efecto aleatorio explica la variabilidad en la variable dependiente. Por lo tanto, para que un efecto pueda considerarse aleatorio, es necesario que la variable dependiente presente cierta variabilidad no explicada asociada con las unidades del efecto aleatorio.

La siguiente tabla puede ayudar ayudar a determinar si un efecto es fijo o aleatorio

1. ¿Cuál es el número de niveles?

Pequeño	Fijo
Grande o infinito	Posiblemente aleatorio

2. ¿Son los niveles repetibles?

Sí	Fijo
No	Aleatorio

3. ¿Hay, conceptualmente, un número infinito de niveles?

No	Posiblemente fijo
Sí	Posiblemente aleatorio

4. ¿Se necesitan realizar inferencias para niveles no incluidos en el muestreo?

No	Posiblemente fijo
Sí	Posiblemente aleatorio

Por ejemplo, en un estudio sobre satisfacción en el trabajo (variable dependiente) de los empleados (unidades observadas) de un cierto número de empresas (efecto aleatorio), si el nivel de satisfacción de los empleados de unas empresas es mayor que el de otras y el investigador no lo tiene en cuenta, habrá una cierta variabilidad residual asociada con el efecto 'empresa'. Si esta variabilidad fuera próxima a cero, no sería necesario incluir el efecto aleatorio asociado con la empresa.

¿Por qué hay que utilizar modelos mixtos?

Cuando las observaciones están agrupadas en niveles o siguen una cierta jerarquía, las unidades se ven afectados por el grupo al que pertenecen. Las jerarquías (o niveles) permiten representar la relación de dependencia entre los individuos y los grupos a los que pertenecen. Los alumnos que están en una misma escuela se parecen más entre sí que si se hubieran seleccionado aleatoriamente de entre toda la población de alumnos. Los modelos mixtos permiten tener en cuenta que las observaciones no son independientes.

18.2. Formulación del modelo con efectos aleatorios o modelos mixtos

El nombre de *modelos mixtos lineales* viene del hecho de que estos modelos son lineales en los parámetros, y en las covariables, y pueden implicar efectos fijos o aleatorios. Son, por lo tanto, una extensión de los modelos lineales de regresión.

18.2.1. Formulación general

La formulación general de un modelo mixto tiene la siguiente forma:

$$y = X\beta + Zu + \epsilon, \quad u \sim N(0, G), \quad \epsilon \sim N(0, R)$$

donde

- X es una matriz $n \times k$ (k es el número de efectos fijos)
- Z es una matriz $n \times p$ (p es el número de efectos aleatorios)
- β es el vector de efectos fijos y u el de efectos aleatorios
- G es la matriz de varianzas-covarianzas de los efectos aleatorios, con dimensión $p \times p$
- R es la matriz de varianzas-covarianzas del error

18.2.1.1. Estimación de β y u

Se hace mediante las llamadas **ecuaciones de Henderson** (Henderson, 1953). Permiten obtener el mejor estimador lineal insesgado de $X\beta$ y el mejor predictor lineal insesgado de u. Se obtienen maximizando la densidad conjunta de y y u:

$$f(y,u) = f(y|u)f(u), \quad y|u \sim N(X\beta + Zu, R) \quad u \sim N(0, G).$$

Derivando con respecto a β y u se obtienen las ecuaciones de Henderson cuya solución es:

$$\begin{aligned} \hat{\beta} &= \left(X'V^{-1}X\right)^{-1} X'V^{-1}y \\ \hat{u} &= GZ'V^{-1}(y - X\hat{\beta}), \end{aligned}$$

donde $V = ZGZ' + R$. Pero, V depende de los parámetros de la varianza en el modelo que forman parte de G y R y que es necesario estimar, como se muestra a continuación.

18.2.1.2. Estimación de los componentes de la varianza

Los métodos más comunes para la estimación de los parámetros de las matrices de covarianza son: Máxima verosimilitud (ML) o Máxima verosimilitid restringida (REML). No existe una solución cerrada para los estimadores, y se hace de forma numérica o mediante algoritmos iterativos. REML tiene en cuenta los grados de libertad utilizados para estimar los efectos fijos en el modelo. Si n es pequeño, REML dará mejores estimaciones que ML, si n es grande, no habrá prácticamente ninguna diferencia. El método preferido es REML.

18.2.2. Inferencia y selección del modelo

18.2.2.1. Contrastes de hipótesis para los efectos fijos, β

Utilizando la distribución aproximada:

$$\hat{\beta} \sim N\left(\beta, \underbrace{(X'\hat{V}^{-1}X)^{-1}}_{Var(\hat{\beta})}\right)$$

- Si se contrastan parámetros individuales se acudirá al t-test para un solo efecto,

- Si se contrastan un conjunto de parámetros se acudirá al F-test para más de un efecto,
- También se pueden comparar modelos usando el test de la razón de verosimilitud, LRT por sus siglas en inglés (ojo! hay que utilizar ML para estimar los parámetros de la varianza)

$$LRT = -2\left[ln(l_{H_0}) - ln(l_{H_1})\right] \approx \chi^2_{df}. \tag{18.1}$$

18.2.2.2. Contrastes de hipótesis para los parámetros de varianza

Al usar el test LRT (18.1) se ha de tener en cuenta que la distribución asintótica del estadístico del test depende de si el valor del parámetro bajo la hipótesis nula (H_0) está en la frontera del espacio paramétrico (es decir si se está testando si el parámetro de varianza es cero o no):

- **Caso 1**: El valor de los parámetros de varianza bajo la H_0 no están en la frontera del espacio paramétrico (por ejemplo, al contrastar que los parámetros de varianza de dos efectos aleatorios son iguales o no). En ese caso se utiliza el test normalmente.

- **Caso 2**: El valor de los parámetros de varianza bajo la H_0 están en la frontera del espacio paramétrico (por ejemplo, si se quiere contrastar que la varianza del efecto aleatorio es cero o no). La distribución asintótica del estadístico del test es una mixtura entre χ^2_p y χ^2_{p-1}, concretamente $0{,}5\chi^2_p + 0{,}5\chi^2_{p-1}$, donde p es el número de parámetros de la varianza que se hacen cero bajo la H_0.

18.2.3. Diagnosis del modelo

En el caso de modelos mixtos se ha de verificar la hipótesis de normalidad tanto para los residuos al nivel más bajo (medidas repetidas en los raíles) como para los efectos aleatorios (raíles), y también las de independencia (en su caso).

En el caso de los modelos mixtos, se utilizan los residuos condicionales, estos son la diferencia entre los valores observados y el valor predicho condicional:

$$\hat{\epsilon} = y - X\hat{\beta} - Z\hat{u}.$$

Estos residuos tienden a estar correlados y sus varianzas pueden cambiar de un grupo a otro, aunque en el verdadero modelo los residuos sean incorrelados y con varianza constante. Para solucionar este problema se pueden escalar los residuos por sus desviaciones estándar (o las estimaciones de éstas), dando lugar a los *residuos estandarizados* (si las desviaciones estándar son conocidas), o a los *residuos studentizados* (si son desconocidas y se utilizan estimaciones de las mismas). Con estos residuos se hace un análisis similar al caso de modelos de regresión lineal.

Además se tendrá que comprobar la hipótesis de normalidad de los efectos aleatorios

18.3. Funciones de R para ajustar modelos mixtos

Hay varios paquetes de R para el ajuste de modelos mixtos. Los más usados son `nlme` y `lme4`. El segundo es una versión del primero que incluye modelos más generales, y mejora los gráficos. A continuación se describe la función principal del paquete `lme4`.

18.3.1. La función `lmer()`

Esta función permite el uso de efectos aleatorios anidados y de errores correlados o heterocedásticos dentro de los grupos. En general para definir un modelo mixto se necesita especificar la estructura de la media y de la parte aleatoria del modelo, incluidos los factores de agrupamiento, así como la estructura de correlación (si la hay).

También se puede especificar el método de estimación: ''REML" o ''ML".

La parte aleatoria del modelo se incluye entre paréntesis en la ecuación y | separa las variables de agrupamiento de las predictoras, si no hay variables predictoras para la parte aleatoria se pondría un 1.

La función `VarCorr()` aplicada a un objeto `lmer` da información sobre la estructura de componentes de varianza.

18.4. Caso práctico

En esta sección se verá como construir diferentes modelos con efectos aleatorios según a qué nivel estén medidas las variables explicativas y se terminará dando una guía sobre como construir estos modelos en la práctica. Los datos con los que se va a trabajar se encuentran en el dataframe `Hsb82` del paquete `mlmRev` y provienen de un estudio titulado *High School and Beyond*. Los datos corresponden a 7185 estudiantes repartidos en 160 escuelas, el número de alumnos por escuela varía entre 14 y 67. La variable de interés, `mAch` es el nivel estandarizado alcanzado en matemáticas. Una cuestión inicial que se puede plantear es si el nivel socioeconómico (`cses`) del alumno predice las diferencias en el nivel de matemáticas. Para ello se ajustaría el modelo:

$$y_j = \beta_0 + \beta_1 x_j + \epsilon_j,$$

este modelo ignora que los alumnos provienen de distintos centros (por eso solo aparece el subíndice j que es el que representa a las unidades de nivel más bajo, en este caso a los alumnos).

```
library(mlmRev)
multi0 <- lm(mAch ~ cses, data = Hsb82)
summary(multi0)
#>
#> Call:
#> lm(formula = mAch ~ cses, data = Hsb82)
#>
```

```
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -17.8660  -5.1165   0.2966   5.3880  14.8705
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 12.74785    0.07933  160.69   <2e-16 ***
#> cses         2.19117    0.12010   18.24   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 6.725 on 7183 degrees of freedom
#> Multiple R-squared:  0.04429,    Adjusted R-squared:  0.04415
#> F-statistic: 332.8 on 1 and 7183 DF,  p-value: < 2.2e-16
```

La ordenada en el origen es 12.75 y la pendiente 2.19, lo que indica que por cada unidad que aumenta el nivel socio-económico, la puntuación del test aumenta en 2.19 unidades, además se puede ver que el coeficiente es significativo.

Supóngase que ocurre esta situación:

Los alumnos de la escuela A sacan, en promedio, mejores notas que las que le asignaría el modelo ajustado, con la escuela B ocurre lo contrario. El gráfico indica que la ordenada en el origen no debería ser la misma para todos los centros, sino que debería ser distinta para distintos centros. Es decir, el valor predicho debe ajustarse hacia arriba o abajo, eso se puede conseguir permitiendo que cada escuela tenga su propia ordenada en el origen:

$$y_{ij} = \beta_{0i} + \beta_1 x_{ij} + \epsilon_{ij}$$

Este modelo es similar al anterior añadiendo el subíndice i para identificar el centro al que pertenece cada alumno. En realidad, se utiliza una variable categórica con tantas categorías como escuelas.

```
multi1 <- lm(mAch ~ cses + school, data = Hsb82)
```

Se está considerando a las escuelas como un efecto fijo y no aleatorio, es decir, implícitamente se está suponiendo que solo interesan estas escuelas en particular.

Las situación se pueden complicar más, es posible que el efecto del nivel socio-económico sea distinto para cada centro, es decir, que un aumento de una unidad en ese nivel puede dar lugar a un aumento distinto en la nota del test en cada centro. En la Figura 18.1 se ve como la pendiente de la recta para la escuela C es distinta a las dos anteriores.

Figura 18.1: Ilustración de posibles escenarios para tres escuelas

El modelo que permite tener en cuenta esta situación es:

$$y_{ij} = \beta_{0i} + \beta_{1i}x_{ij} + \epsilon_{ij}$$

donde aparece ahora el sub-índice i también en la pendiente, lo que indica que cada centro tiene una pendiente diferente.

```
multi2 <- lm(mAch ~ cses * school, data = Hsb82)
```

El código anterior generaría 159 coeficientes más que son los que se incluirían con la interacción. Pero no interesan en estas escuelas en concreto, sino la población de la que estas escuelas son una muestra.

Con un modelo con efectos aleatorios se pueden contestar a preguntas como: ¿Cuales son las causas de esta variabilidad? ¿Qué variables pueden explicarla?

18.4.1. Modelo con ordenada en el origen aleatoria

Es el modelo mixto más sencillo. Se considera que los datos tienen una estructura con dos niveles, los alumnos están en el nivel 1 y están agrupados en escuelas, nivel 2. Se va a empezar

suponiendo que no se dispone de ninguna variable explicativa, y que por lo tanto nuestro único interés es la diferencia entre las notas medias del test de matemáticas entre los distintos centros.

Los dos niveles del modelo son:

$$\text{Nivel 1: } y_{ij} = \mu_i + \epsilon_{ij}$$

- El subíndice j corresponde a individuos y el i a escuelas, si se considera a las escuelas como un efecto aleatorio,
- β_{0i} (la media de cada escuela) vendría dada por:

$$\text{Nivel 2: } \mu_i = \beta_0 + u_i,$$

- β_0 es la media de todos los alumnos,
- u_i es la desviación de la media de la escuela i de la media de todas las escuelas.

Poniendo las dos ecuaciones juntas:

$$y_{ij} = \beta_0 + u_i + \epsilon_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots n_m \tag{18.2}$$

- La media de y para el grupo i es $\beta_0 + u_i$,
- Los residuos a nivel individual ϵ_{ij} son la diferencia entre el valor de la variable respuesta del individuo j y la media del grupo al que pertenece,
- $u_i \sim N(0, \sigma_i^2)$, $\epsilon_{ij} \sim N(0, \sigma^2)$, y ambos son independientes, es decir, las observaciones que provienen de distintas escuelas son independientes.

En el ejemplo de las escuelas:

```
library(lme4)
Modelo0 <- lmer(mAch ~ 1 + (1 | school), data = Hsb82)
Modelo0
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: mAch ~ 1 + (1 | school)
#>    Data: Hsb82
#> REML criterion at convergence: 47116.79
#> Random effects:
#>  Groups   Name        Std.Dev.
#>  school   (Intercept) 2.935
#>  Residual             6.257
#> Number of obs: 7185, groups:  school, 160
#> Fixed Effects:
#> (Intercept)
#>       12.64
```

- La media total estimada es 12.64,
- La media para la escuela i es: $12.64 + \hat{u}_i$, donde $\hat{u}_i$ es el efecto aleatorio de la escuela.

Para obtener los valores predichos de los efectos aleatorios se utiliza la función `ranef()`.

El siguiente gráfico permite ver los efectos aleatorios junto con sus intervalos de confianza (las escuelas han sido ordenadas atendiendo a su media para apreciar mejor la variabilidad entre las mismas).

Se dibujan los efectos aleatorios para ver si siguen una distribución Normal, para eso se ha de ajustar el modelo con la función `lmer()`:

```
library(lattice)
qqmath(ranef(Modelo0, condVar = TRUE))$school
```

Una primera aproximación para contrastar si hay o no diferencias entre los grupos sería calcular el intervalo de confianza para σ_u:

```
confint(Modelo0)
#>                  2.5%    97.5%
#> .sig01       2.594729  3.315880
#> .sigma       6.154803  6.361786
#> (Intercept) 12.156289 13.117121
```

el intervalo no contiene al cero, pero la forma más correcta de hacerlo sería utilizando el test LRT (18.1) para:

$$\begin{aligned} H_0: &\quad \sigma_u^2 = 0 \Rightarrow y_{ij} = \beta_0 + \epsilon_{ij} \\ H_1: &\quad \sigma_u^2 \neq 0 \Rightarrow y_{ij} = \beta_0 + u_i + \epsilon_{ij}. \end{aligned}$$

El resultado del test en este caso se compara con el valor de una mixtura de distribuciones Chi-cuadrado $0{,}5\chi_0^2 + 0{,}5\chi_1^2$.

```
Modelo_NULL <- lm(mAch ~ 1, data = Hsb82)
test <- -2 * logLik(Modelo_NULL, REML = T) + 2 * logLik(Modelo0, REML = T)
mean(pchisq(test, df = c(0, 1), lower.tail = F))
#> [1] 9.320673e-217
```

Conclusión: el efecto aleatorio es necesario en el modelo.

El siguiente paso sería introducir las variables explicativas, ya estén al nivel 1 o al 2.

18.4.1.1. Variables explicativas en el Nivel 1 (individuos)

Como la variable explicativa está medida al Nivel 1, se introduce en la ecuación del Nivel 1:

- Nivel 1: $y_{ij} = \mu_i + \beta_1 x_{ij} + \epsilon_{ij}$
- Nivel 2: $\mu_i = \beta_0 + u_i$

Si x es una variable continua, este modelo asume que la pendiente de la recta es la misma para todas las escuelas (por eso β_1 no lleva el subíndice i). Poniendo las dos ecuaciones juntas:

$$y_{ij} = \underbrace{\beta_0 + \beta_1 x_{ij}}_{\text{efectos fijos}} + \underbrace{u_i + \epsilon_{ij}}_{\text{efectos aleatorios}}$$

En este modelo, la relación global entre y y x viene representada por la línea recta con ordenada en el origen β_0 y pendiente β_1. Sin embargo, la ordenada en el origen para una determinada escuela i viene dada por $\beta_0 + u_i$. Será mayor o menor que que la ordenada en el origen global β_0 por una cantidad u_i. Aunque la ordenada en el origen varía de grupo a grupo, la pendiente es la misma para todos los grupos. Todas las líneas rectas ajustadas para cada grupo son paralelas.

En el ejemplo de las escuelas, se introduce como variable explicativa `cses` (nivel socioeconómico centrado):

```
Modelo1 <- lmer(mAch ~ cses + (1 | school), data = Hsb82)
Modelo1
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: mAch ~ cses + (1 | school)
#>    Data: Hsb82
#> REML criterion at convergence: 46724
#> Random effects:
#>  Groups   Name        Std.Dev.
#>  school   (Intercept) 2.945
#>  Residual             6.084
#> Number of obs: 7185, groups:  school, 160
#> Fixed Effects:
#> (Intercept)         cses
#>      12.636        2.191
```

Ahora se tienen dos efectos fijos:

$$\hat{\beta}_0 = 12{,}64$$
$$\hat{\beta}_1 = 2{,}19$$

$\hat{\beta}_0$ es la nota media para alumnos con nivel socieconómico medio (la variable está centrada). La recta media vendría dada por:

$$E[y|cses] = 12{,}64 + 2{,}19\ cses$$

Para comprobar si la variable `cses` es significativa se utiliza el test LRT (18.1). Primero se tienen que ajustar de nuevo los modelos que se quieren comparar usando máxima verosimilitud (en vez de REML). Si se utiliza la función `lmer()` para ajustar el modelo no es necesario reajustar con ML pues la función anova lo hará automáticamente, mientras que si se usa la función `lme()` sí será necesario hacerlo.

```
anova(Modelo0, Modelo1)
#> Data: Hsb82
#> Models:
#> Modelo0: mAch ~ 1 + (1 | school)
#> Modelo1: mAch ~ cses + (1 | school)
#>         npar   AIC   BIC logLik deviance Chisq Df Pr(>Chisq)
#> Modelo0    3 47122 47142 -23558    47116
#> Modelo1    4 46728 46756 -23360    46720 395.4  1  < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Por lo tanto, el nivel socioeconómico afecta a los resultados escolares. Comparado con el modelo sin la variable explicativa (Modelo0), la inclusión del nivel socio-económico (Modelo1), ha reducido la variabilidad a nivel del alumno en un 2.8 % ((6.257 - 6.084)/6.257 = 0.028)

18.4.1.2. Variables explicativas en el Nivel 2 (grupos)

Si las variable explicativas son medidas al Nivel 2:

$$\begin{aligned} \text{Nivel 1:} \quad & y_{ij} = \mu_i + \epsilon_{ij} \\ \text{Nivel 2:} \quad & \mu_i = \beta_0 + \beta_2 s_i + u_i \end{aligned}$$

$$y_{ij} = \underbrace{\beta_0 + \beta_2 s_i}_{\text{efectos fijos}} + \underbrace{u_i + \epsilon_{ij}}_{\text{efectos aleatorios}}$$

En nuestro ejemplo, la variable utilizada es `sector` (público o privado):

$$mAch = \beta_0 + \beta_2\ sector + u_i + \epsilon_{ij}$$

Se ajusta el modelo usando la función `lmer()`

```
Modelo2 <- lmer(mAch ~ sector + (1 | school), data = Hsb82)
Modelo2
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: mAch ~ sector + (1 | school)
#>    Data: Hsb82
#> REML criterion at convergence: 47080.13
#> Random effects:
#>  Groups   Name        Std.Dev.
#>  school   (Intercept) 2.584
#>  Residual             6.257
#> Number of obs: 7185, groups:  school, 160
#> Fixed Effects:
#>    (Intercept)  sectorCatholic
#>         11.393           2.805
```

$$E[y|sector] = 11{,}39 + 2{,}8\ sector$$

o equivalentemente

$$E[y|sector = 0] = 11{,}39$$
$$E[y|sector = 1] = 11{,}39 + 2{,}8 = 14{,}19$$

La nota de un alumno en una escuela privada se espera que sea 2.8 unidades mayor que la de un alumno en una escuela pública (se puede generalizar puse se asume que las escuelas son un efecto aleatorio). La varianza del efecto aleatorio de nivel 2 σ_u^2 ha descendido:$(2{,}935^2 - 2{,}584^2)/2{,}935^2 = 0{,}22$, es decir que se ha reducido en un 22 % la variabilidad no explicada entre los centros al introducir la variable sector.

Para contrastar si la variable sector es significativa se usa de nuevo el test LRT:

```
anova(Modelo0, Modelo2)
#> Data: Hsb82
#> Models:
#> Modelo0: mAch ~ 1 + (1 | school)
#> Modelo2: mAch ~ sector + (1 | school)
#>         npar   AIC   BIC logLik deviance  Chisq Df Pr(>Chisq)
#> Modelo0    3 47122 47142 -23558    47116
#> Modelo2    4 47087 47115 -23540    47079 36.705  1  1.374e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Por lo tanto, el hecho de que la escuela sea pública o privada influye en el resultado académico de los alumnos.

18.4.2. Modelo con pendiente aleatoria

En este tipo de modelos se supone que la relación entre la variable respuesta y las variables explicativas va a ser distinta para las distintas unidades de nivel 2, es decir, la relación puede cambiar de un centro a otro. Por ejemplo, el efecto del nivel socioeconómico en las notas puede ser distinto en distintos centros, de modo que se puede relajar el modelo anterior, en el que la pendiente era la misma para todos los grupos, permitiendo que la pendiente varíe aleatoriamente entre los grupos.

$$\begin{array}{ll} \text{Nivel 1:} & y_{ij} = \mu_i + \beta_{1i} x_{ij} + \epsilon_{ij} \\ \text{Nivel 2:} & \mu_i = \beta_0 + u_i \\ & \beta_{1i} = \beta_1 + v_i \end{array}$$

Poniendo las dos ecuaciones juntas:

$$y_{ij} = \underbrace{\beta_0 + \beta_1 x_{ij}}_{\text{efectos fijos}} + \underbrace{u_i + v_i x_{ij} + \epsilon_{ij}}_{\text{efectos aleatorios}}, \quad \begin{pmatrix} u_i \\ v_i \end{pmatrix} \sim N(0, G_i) \quad G_i = \begin{pmatrix} \sigma_u^2 & \\ \sigma_{uv} & \sigma_v^2 \end{pmatrix},$$

donde σ_{uv} es la covarianza entre las ordenadas en el origen de los grupos y las pendientes. Un valor positivo de la covarianza implica que grupos con un valor del efecto de grupo u_i elevado, tienden a tener valores elevados de v_i, o equivalentemente, centros con ordenada en el origen alta, tienen pendiente alta.

El modelo en `R` sería:

```
Modelo3 <- lmer(mAch ~ cses + (cses | school), data = Hsb82)
Modelo3
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: mAch ~ cses + (cses | school)
#>    Data: Hsb82
#> REML criterion at convergence: 46714.23
#> Random effects:
#>  Groups   Name        Std.Dev. Corr
#>  school   (Intercept) 2.9464
#>           cses        0.8331   0.02
#>  Residual             6.0581
#> Number of obs: 7185, groups:  school, 160
#> Fixed Effects:
#> (Intercept)         cses
#>      12.636        2.193
```

El efecto del nivel socieconómico en la escuela i se estima como $2{,}19 + \hat{u}_i$, y la varianza de las pendientes entre escuelas es $0{,}833^2 = 0{,}694$. Para la *escuela promedio* se predice un aumento de 2,19 en la puntuación cuando el nivel socioeconómico aumenta en una unidad.

Ahora se tienen los siguientes parámetros de la varianza:

$$\hat{\sigma}_u^2 = 8{,}68 \quad \hat{\sigma}_v^2 = 0{,}694 \quad \hat{\sigma}_{uv} = \rho \sigma_u \sigma_v = 0{,}051 \quad \hat{\sigma}^2 = 36{,}7$$

La varianza de ordenada en el origen estimada, 8,68 se interpreta como la variabilidad entre las escuelas para un nivel socioeconómico medio.

El parámetro de covarianza estimado es $\sigma_{uv} = 0{,}051$, por lo que se puede plantear si es necesario o no.

Para comprobarlo, el contraste de hipótesis sería: En este caso

$$H_0 : \sigma_{uv} = 0 \quad \text{y} \quad H_1 : \sigma_{uv} \neq 0$$

```
Modelo3.1 <- lmer(ses ~ cses + (cses || school), data = Hsb82)
```

Cuando se quiere que haya un efecto aleatorio para la ordenada en el origen y para la pendiente pero que sean incorrelados en la función solo hay que poner doble barra en vez de simple.

En este caso no es necesario utilizar la mixtura de χ^2 para $H_0 : \sigma_{uv} = 0$, pues σ_{uv} puede tomar cualquier valor.

```
anova(Modelo3.1, Modelo3)
#> Data: Hsb82
#> Models:
#> Modelo3.1: ses ~ cses + ((1 | school) + (0 + cses | school))
#> Modelo3: mAch ~ cses + (cses | school)
#>           npar     AIC     BIC logLik deviance Chisq Df Pr(>Chisq)
#> Modelo3.1    5 -226164 -226129 113087  -226174
#> Modelo3      6   46723   46764 -23355    46711     0  1          1
```

Por lo tanto se puede suponer que la covarianza es 0.

El siguiente paso sería contrastar si es necesario que las rectas tengan pendientes diferentes, es decir, H_0: $\sigma_v^2 = 0$, H_1: $\sigma_v^2 > 0$, en este caso sí se necesita la aproximación:

```
test <- -2 * logLik(Modelo1, REML = T) +
  2 * logLik(Modelo3.1, REML = T)
mean(pchisq(test, df = c(0, 1), lower.tail = F))
#> [1] 0
```

Por lo tanto la pendiente es diferente en las distintas escuelas.

Además, se puede usar algún criterio de información para comparar los modelos:

```
AIC(logLik(Modelo3))
#> [1] 46726.23
AIC(logLik(Modelo3.1))
#> [1] -226113.6
AIC(logLik(Modelo1))
#> [1] 46732
```

A veces la covariable media a nivel 2 (a nivel de grupo, en este caso escuelas) puede explicar tanto la variabilidad de la ordenada en el origen como la pendiente:

$$\begin{array}{ll} \text{Nivel 1:} & y_{ij} = \mu_i + \beta_{1i} x_{ij} + \epsilon_{ij} \\ \text{Nivel 2:} & \mu_i = \beta_0 + \beta_{2i} s_i + u_i \\ & \beta_{1i} = \beta_1 + \beta_{3i} s_i + v_i \end{array}$$

$$y_{ij} = \underbrace{\beta_0 + \beta_1 x_{ij} \beta_2 s_i + \beta_3 x_{ij} : s_i}_{\text{efectos fijos}} + \underbrace{u_i + v_i x_{ij} + \epsilon_{ij}}_{\text{efectos aleatorios}}$$

Al introducir la variable medida al nivel 2, la parte fija se modifica (con respecto al Modelo 3) pero no la parte aleatoria:

- β_2 indica si los centros privados son diferentes de los públicos en cuanto a su nota media.
- β_3 indica si los centros privados difieren de los públicos en cuanto a la relación entre el nivel socio-económico y la puntuación.

```
Modelo4 <- lmer(mAch ~ cses * sector + (cses || school),
  data = Hsb82
)
Modelo4
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: mAch ~ cses * sector + ((1 | school) + (0 + cses | school))
#>    Data: Hsb82
#> REML criterion at convergence: 46648.85
#> Random effects:
#>  Groups   Name        Std.Dev.
#>  school   (Intercept) 2.5971
#>  school.1 cses        0.5182
#>  Residual             6.0580
#> Number of obs: 7185, groups:  school, 160
#> Fixed Effects:
#>         (Intercept)                 cses       sectorCatholic
#>              11.393                2.784                2.805
#> cses:sectorCatholic
#>              -1.346
```

Los centros privados tienen una nota media más alta que los públicos (2.81), y tienen una pendiente más suave que la de los centros públicos (-1.35). En un colegio privado la mejora de la nota con respecto al nivel socio-económico es más lenta que un colegio público.

18.4.3. ¿Cómo construir el modelo en la práctica?

1. Se ajusta el modelo con todos los efectos fijos y aleatorios posibles

```
Modelo5 <- lmer(mAch ~ cses * sector + (cses | school), data = Hsb82)
```

2. Se contrasta qué efectos aleatorios son significativos, sin mover los efectos fijos

Primero se ve si la covarianza entre efectos fijos y aleatorios es cero o no:

```
# Se ajusta el modelo con covarianza = 0
Modelo5.1 <- lmer(ses ~ cses * sector + (cses || school),
  data = Hsb82
)
anova(Modelo5.1, Modelo5)
#> Data: Hsb82
#> Models:
#> Modelo5.1: ses ~ cses * sector + ((1 | school) + (0 + cses | school))
#> Modelo5: mAch ~ cses * sector + (cses | school)
#>           npar     AIC     BIC logLik deviance Chisq Df Pr(>Chisq)
#> Modelo5.1    7 -220069 -220021 110042  -220083
#> Modelo5      8   46650   46705 -23317    46634     0  1          1
```

Como lo es, no se tendría que contrastar nada más, los efectos aleatorios son los que se han incluido en el Modelo5. Si no hubiera sido significativa se continuaría testando si la pendiente aleatoria es significativa y si la ordenada en el origen lo es.

3. Una vez elegidos los efectos aleatorios (que se mantienen), se eligen los efectos fijos:

```
Modelo6 <- update(Modelo5, . ~ . - cses:sector)
anova(Modelo6, Modelo5)
#> Data: Hsb82
#> Models:
#> Modelo6: mAch ~ cses + sector + (cses | school)
#> Modelo5: mAch ~ cses * sector + (cses | school)
#>         npar   AIC   BIC logLik deviance  Chisq Df Pr(>Chisq)
#> Modelo6    7 46678 46726 -23332    46664
#> Modelo5    8 46650 46705 -23317    46634 29.983  1  4.358e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Como la interacción es significativa ya no se tendría que hacer ningún test más y este sería el modelo final

Resumen

En este capítulo se han introducido los modelos mixtos o modelos con efectos aleatorios, en particular:

- Se han dado las claves para distinguir entre efectos fijos y aleatorios.
- Se ha presentado la formulación del modelo y se han dado las claves para llevar a cabo la estimación del mismo.
- Se han explicado las etapas del proceso a seguir para el ajuste de este tipo de modelos.
- Se ha mostrado el uso de `R` para ajustar estos modelos.
- Se ha ilustrado el análisis de modelos multinivel como caso particular de un modelo con efectos aleatorios.

Capítulo 19

Modelos sparse y métodos penalizados de regresión

María Durbán

19.1. Introducción

El modelo de regresión lineal múltiple: $y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \varepsilon$, 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 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 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 lo tanto, 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 serían:

- **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. Esto es lo que se llama *Componentes principales* que se desarrolla en el Capítulo 11.

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

19.2. 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 calculados 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 $k = 1, 2, \ldots, p$ se calcula 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 es mayor.
4. Elegir entre los modelos: $M_1, \ldots, M_p$ aquel que es mejor utilizando una criterio como AIC, BIC o R^2 ajustado.

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

19.2.1. Ejemplo: Sueldo de jugadores de béisbol

Se va a aplicar el método descrito al conjunto de datos `Hitters` del paquete `ISRL2`. 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 alguno de los jugadores, se pueden identificar utilizando la función `is.na()`. Y la función `sum()` permite ver cuántas 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.

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).

En este caso, 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
```

19.3. 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 sobre-ajuste. 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.

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

19.3.1. 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 $k = 0, 1, 2, \ldots, p$:
   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}. Donde **mejor** significa tener el RSS más bajo o el R^2 más alto
3. Se elige entre los modelos: $M_0, \ldots, M_p$ aquel que es mejor utilizando una 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.

19.3.2. 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 $k = p, p-1, \ldots, 1$:
   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}. Donde **mejor** significa tener el RSS más bajo o el R^2 más alto
3. Se elige entre los modelos: $M_0, \ldots, M_p$ 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 $n < p$

19.3.3. Ejemplo: Sueldo de jugadores de béisbol

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
#> -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)
```

Usando `regsubsets()` en la muestra de entrenamiento:

```
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 testeo. 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
```

19.4. 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.

19.4.1. Regresión ridge

Se recuerda que el ajuste por mínimos cuadrados estima $\beta_0, \beta_1, \ldots, \beta_p$ 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:

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

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 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, y la cantidad que fuerza a esos coeficientes a contraerse hacia cero está determinada por λ, el *parámetro de tuneado*. Si $\lambda = 0$ se está en el caso de mínimos cuadrados, y cuanto mayor sea λ, mayor será el precio a pagar para que esos coeficientes sean distintos de 0. Si λ es extremadamente grande los coeficientes estarán muy próximos a 0 para poder hacer este 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.

19.4.1.1. 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), esto quiere 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.

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

Con esto se consigue que los coeficientes sean comparables.

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 son 0, 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 es de interés mantener todas en el modelo. En estos casos, la regresión ridge soluciona los problemas de multicolinealidad.

19.4.1.2. Ajuste de regresión ridge en R

El paquete que se va a usar para regresión ridge (y para otros métodos 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 calcular el ajuste 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 100 columnas (una para 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 $\lambda = 11{,}498$, y su suma al cuadrado, $\sum_{j=1}^{p} \beta_j^2$:

```
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:

```
plot(ridge.mod, xvar = "lambda", label = TRUE)
```

El lado izquierdo del gráfico 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 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 y 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 va a dividir los datos en una muestra de entrenamiento y otra de testeo 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 lambda (por ejemplo $\lambda = 4$), y se evalúa su MSE en la muestra de testeo. Para eso se usará la función `predict()`. En este caso, para obtener las predicciones para la muestra de testeo, se reemplaza `type = çoefficients"` 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 ve 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 dicho parámetro de una forma automática.

19.4.2. Selección del parámetro de tuneado

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

Una buena manera de elegir λ es usar validación cruzada, 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.
- Se repite tomando como muestra de testeo 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, 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:

[1] además se ha de añadir 'exact = T' en la función 'predict()'

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

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

En este gráfico 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 menor cuanto mayor se 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 corresponde al MSE más una desviación típica.

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
```

Esto representa una mejora sobre el error de predicción que se había obtenido cuando $\lambda = 4$.

19.4.3. 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, por lo tanto **la regresión ridge es útil cuando la mayoría de las variables predictoras son útiles**.

La regresión *lasso* es una alternativa a la regresión ridge cuyo objetivo es precisamente eliminar esa desventaja de la regresión ridge, fue introducida por Tibshirani (1996). **La regresión lasso es útil cuando las mayoría de las variables predictoras no son útiles**. 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 cuando λ es suficientemente grande los coeficientes serán 0, de modo que se estará haciendo 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**.

¿Por qué lasso hace que los coeficiente se contraigan exactamente hacia cero? Para entenderlo se va a ver una formulación equivalente 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 esta 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 tengo un *presupuesto* s, habrá un λ que corresponda en la formulación previa que corresponda al mismo problema y viceversa. Supóngase que se hacen mínimos cuadrados y se obtienen unos ciertos parámetros estimados, y supóngase que la suma de los valores absolutos de los coeficientes 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 los 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 mi *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 Figura 19.1 (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, $\boldsymbol{\beta}$, es donde se alcanzaría el valor mínimo de los mínimos cuadrados (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 parámetro va a ser 0. En el caso de lasso la región es un diamante, y por lo tanto tiene vértices, en la Figura el contorno toca a la región de restricción en el caso en que $\beta_1 = 0$.

Figura 19.1: Contornos (rojo) de RSS y regiones de restricción (en azul) para lasso (izquierda) y ridge (derecha).

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)
```

Se puede ver que dependiendo del valor del parámetro de tuneado, algunos de los coeficientes se hacen exactamente 0. Ahora se va a elegir mediante validación cruzada y se calcula el error de predicción:

```
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 testeo en el caso de mínimos cuadrados (224669,8), y bastante parecido al obtenido con regresión ridge (cuando el parámetro de tuneado se elige mediante validación cruzada, 139856,6).

Sin embargo, lasso tiene una ventaja importante con respecto a la regresión ridge ya que los coeficientes estimados son *sparse*. Aquí se ve 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
```

19.4.4. 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 p variables, aunque haya más de p relevantes; 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 combinan 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 (eso es lo que se ha hecho para usar regresión ridge o lasso). 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)
```

En el gráfico se ve cómo 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 representa los valores de α.

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

```
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 ve 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 las variables 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 han introducido técnicas para mejorar la predicción y la interpretabilidad de los modelos de regresión, en particular:

- Se ha mostrado el uso de la técnica de selección del mejor subconjunto de variables en el modelo, así como los métodos *stepwise*.
- Se han presentado 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 ha mostrado como seleccionar los parámetros de tuneado que controlan la regresión penalizada.
- Se ha ilustrado el uso de todas las metodologías propuesta en el capítulo mediante el análisis de un caso práctico.

Capítulo 20

Modelización de series temporales

Mª Carmen García Centeno

20.1. Conceptos básicos

El análisis de series temporales es muy útil para analizar tendencias y comportamiento de datos. Con dicho análisis, se tratará de obtener modelos que expliquen su dinámica y puedan utilizarse para predecir valores futuros y tomar decisiones de la forma más efectiva posible.

Una serie temporal se puede definir como el conjunto de valores observados, en periodos consecutivos de tiempo de la misma amplitud, correspondientes a distintas variables aleatorias. Algunos conceptos claves sobre los que se fundamenta el análisis de series temporales son los siguientes:

1. **Proceso estocástico**. Es una sucesión de variables aleatorias, de tal forma que, en cada instante del tiempo, un proceso estocástico es una única variable aleatoria con una distribución de probabilidad. Desde el punto de vista práctico, es muy complicado y a veces imposible, caracterizar a 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 ($\mu_t = E(Y_t)$), la varianza ($Var(Y_t) = \sigma_t^2 = E(Y_t - \mu_t)^2$) 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 se cumple que la media y la varianza se mantienen invariantes a lo largo del tiempo y las covarianzas solo dependen del lapso de tiempo que transcurre entre las variables. Es decir, $E(Y_t) = \mu$, $Var(Y_t) = E(Y_t - \mu)^2 = \sigma^2$ y $Cov(Y_t, Y_{t-k}) = E(Y_t - \mu)(Y_{t-k} - \mu) = \gamma(k)$.

3. **Función de autocovarianzas**. Está formada por el conjunto de autocovarianzas calculadas para distintos órdenes (k). Tiene como finalidad determinar las relaciones de dependencia existentes 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 de un proceso estocástico (ACF)**. Está formada por el conjunto de correlaciones calculadas para distintos órdenes (k). De forma genérica, para medir la correlación existente entre Y_t e Y_{t-k}, el coefiente de correlación de orden $\rho(k)$, 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) de la siguiente forma:

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

   La representación gráfica de $\rho(k)$ para k=0,1,2,... se llama *correlograma*.

5. **Función de autocorrelación parcial de un proceso estocástico (PACF)**. Está formada por el conjunto de las correlaciones parciales obtenidas para los distintos valores de k. Así, para dos valores de una serie temporal Y_t e Y_{t-k}, mide la correlación existente entre ellos ajustada del efecto de los periodos intermedios $(Y_{t-1}, Y_{t-2}, ..., Y_{t-(k-1)})$.

6. **Ergodicidad**. Un proceso estocástico es ergódico cuando a partir de un determinado desfase temporal entre las variables, la relación existente entre ellas tiende a desaparecer. Es decir, las covarianzas y el coeficiente de correlación tienden a cero, ($\lim_{k\to\infty} \gamma(k) = 0$ y $\lim_{k\to\infty} \rho(k) = 0$).

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,

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

20.2. Modelos ARIMA

Una de las metodologías utilizadas para modelizar el comportamiento de las series temporales son los modelos ARIMA (AutoRegresive Integrated Moving Average) desarrollados por Box y Jenkins. Estos modelos 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).

En la modelización ARIMA se pueden destacar varias fases. La primera se centra en la **identificación** del modelo ARIMA más adecuado que ha podido generar los datos de la serie temporal. En ella, hay que decidir los órdenes del proceso tanto de la parte regular como de la estacional.

Las funciones de autocorrelación simple (ACF) y parcial (PACF) desempeñan un papel clave en la identificación de estos órdenes. También en esta fase, es necesario comprobar si las series son estacionarias. En el caso de que no lo sean, se realizarán las transformaciones necesarias para convertirla en estacionaria. Las dos razones fundamentales por las cuales puede no ser estacionaria se deben a que la media o la varianza no se mantengan constantes. Si la serie no sea estacionaria en varianza, de las transformaciones posibles de Box-Cox se utilizará la transformación logarítmica. Si la serie no es estacionaria en media, puede ser debido a la existencia de tendencia 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, respectivamente) y en el caso de estacionalidad, se calculará una diferencia estacional como máximo para convertirla en estacionaria.

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 realizarñan, por un lado, los contrastes necesarios que permitan determinar si el modelo propuesto es adecuado o no. Y, por otro, comprobar si los residuos del modelo 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**.

20.3. Análisis de series temporales con R

Las librerías necesarios en la modelización de series temporales con R son:

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

Posteriormente es necesario cargar los datos. Como se ha comentado en capítulos anteriores los datos pueden tener diferente formato (csv, Excel, etc.). En este caso, se utilizarán datos mensuales del INE (www.ine.es) correspondientes al IPC general en el periodo muestral comprendido entre enero de 2002 y marzo de 2022, en formato Excel. La fecha en la que se observan los datos está incluida en la primera columna de este fichero. Los datos se leerán de la siguiente forma:

```
library(CDR)
data(IPC)
```

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

```
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 ejecutará el siguiente código:

```
IPC$Time <- as.Date(IPC$Time)
ggplot(
  data = IPC,
  aes(x = Time, y = ipc)
) +
  geom_line(colour = "blue")
```

Figura 20.1: Evolución del IPC. Perido muestral. Enero 2002 Marzo 2022

También puede ayudar a determinar si la serie es estacionario o no, la descomposición en tendencia, componente estacional estacional, componente cíclico e irregular de la serie del IPC y su representación gráfica.

```
componentes_ts <- decompose(ipc_ts)
autoplot(componentes_ts)
```

Tanto en el gráfico de la serie original del IPC como en su descomposición se aprecia que la serie tiene tendencia y componente estacional, por lo tanto, no es estacionaria en media. Tampoco lo es en varianza.

Figura 20.2: Descomposición aditiva del IPC

Si una serie no es estacionaria ni en media ni en varianza, para evitar problemas de cálculo matemático, es necesario empezar corriguiendo la no estacionariedad en varianza y, posteriormente, la no estacionariedad en media.

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)
) +
  geom_line(colour = "blue")
```

Como se observa en el gráfico del logaritmo del IPC, la serie sigue sin ser estacionaria en media, ya que, tiene estacionalidad y una tendencia creciente. Para correguir la tendencia, se calculará una diferencia regular del logaritmo del IPC ($dlogipc_t = log(ipc_t) - log(ipc_{t-1})$. El código para calcular esta diferencia regulgar y su represenación gráfica es:

```
dlogipc <- diff(logipc, differences = 1)
ts_dlogipc <- data.frame(value = dlogipc, Time = time(dlogipc))
ggplot(
  data = ts_dlogipc,
  aes(x = Time, y = dlogipc)
```

Figura 20.3: Logaritmo del IPC

```
) +
  geom_line(colour = "green")
```

Figura 20.4: Diferencia regular del Logaritmo del IPC

Para correguir la estacionalidad, se calculará una diferencia estacional ($d12logipc_t = log(ipc_t) - log(ipc_{t-12})$. El código para calcular esta diferencia estacional y su represenación gráfica es:

```
d12dlogipc <- diff(dlogipc, 12)
ts_d12dlogipc <- data.frame(value = d12dlogipc, Time = time(d12dlogipc))
ggplot(data = ts_d12dlogipc, aes(x = Time, y = d12dlogipc)) +
```

```
  geom_line() +
  geom_line(colour = "purple")
```

Figura 20.5: Diferencia estacional de la diferencia regular del Logaritmo del IPC

También, para comprobar si la serie es estacionaria en media o no, se puede utilizar el test de raíces unitarias de Dickey-Fuller. La hipótesis nula de este contraste implica la existencia de raíces unitarias y, por lo tanto, que la serie no es estacionaria, mientras que la hipótesis alternativa implica que sí es estacionaria. Si se realiza el test 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, para un nivel de significación del 5 %, 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.

20.3.1. Identificación o especificación del modelo

Obtenida la transformación estacionaria, es necesario, calcular las funciones de autocorrelación simple (ACF) y parcial (PACF) estimadas. A partir de ellas se especificará el modelo más adecuado para la serie del IPC en el periodo muestral analizado.

Los códigos R para ejecutar los comandos que permiten calcular la ACF y PACF de la transformación estacionaria del IPC, con 40 retardos, son respectivamente:

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

Figura 20.6: ACF. Función de autocorrelación simple de d12dLogIPC

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

Conseguida la transformación estacionaria, estas funciones son muy útiles para identificar los ordenes p y q de los procesos puros autorregresivo (AR(p)), de móviles (MA(q)) o mixto ARMA(p,q) de la parte regular. Del mismo modo, se procederá con órdenes P y Q correspondientes a los modelos SAR(P), SMA(Q) o SARMA(P,Q) de la parte estacional (por ejemplo, en el caso de series mensuales, habría que fijarse en los coeficientes correspondientes a los retardos 12, 24, 36, etc. para identificar el modelo de la parte estacional). El comportamiento de estas funciones según el tipo de modelo es el siguiente:

- En un MA(q) o SMA(Q):

Figura 20.7: PACF. Función de autocorrelación parcial de d12dLogIPC

- La función de autocorrelación simple muestral (ACF) se anula para órdenes superiores a q (o de 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 de la ACF muestral).
- La función de autocorrelación parcial muestral (PACF) decrece de forma exponencial o de forma sinusoidal hacía cero.

Ejemplo de la ACF y PAC teóricas, para un proceso estacionario (Yt), 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:

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

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

Este modelo siempre es estacionario y para que sea invertible es necesario que las raíces del polinomio estén fuera del circulo unidad o lo que es lo mismo que $|\theta| < 1$.

La función de autocorrelación (ACF) es:

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

La función de autocorrelación parcial (PACF) se obtiene de la siguiente forma:

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

- En un AR(p) o SAR(P):
  - La ACF decrece rápidamente de forma exponencial o sinusoidal hacia cero.
  - La PACF se anula para órdenes superiores a p (o de 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 si se analiza la PACF muestral).

Ejemplo de la ACF y PAC teóricas, para un proceso estacionario (Yt), 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 y para que sea estacionario es necesario que las raíces del polinomio de retardos estén fuera del circulo unidad o lo que es lo mismo que $|\phi| < 1$.

La función de autocorrelación (ACF) es:

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

La PACF se obtiene de la siguiente forma:

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

- En un ARMA(p,q) o SARMA(P,Q):
  - La ACF tiene un comportamiento irregular en los q (o de Q en el caso estacional) coeficientes. A partir del orden q (o de 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 coeficientes (o de P en el caso estacional). A partir del orden p (o de orden 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 PAC teóricas, para un proceso estacionario (Yt), que sigue un ARMA(1,1).

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

$Y_t = \phi Y_{t-1} + a_t - \theta a_{t-1}$ 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 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 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 función de autocorrelación parcial (FACP) se obtiene:

$$\phi_{11} = \rho_1$$

$$\phi_{22} = \frac{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & \rho_2 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{vmatrix}} = \frac{\rho_2-\rho_1^2}{1-\rho_1^2}$$

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

$$\vdots$$

Tendiendo en cuenta lo anterior, en el caso concreto del IPC, si se analiza el comportamiento de las ACF y PACF muestrales obtenidas para la transformación estacionaria, el modelo ARIMA sería un ARIMA(1,1,0)(0,1,1)12 o SARIMA(1,1,0)(0,1,1).

20.3.2. Estimación del modelo

Para estimar los parámetros del modelo, ya que, el 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 y sus desviaciones típicas correspondiente al modelo propuesto 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
```

20.3.3. Diagnosis, validación y contrastación

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

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

El modelo estimado para el IPC, es estacionario e invertible, ya que, el AR(1) estimado para la parte regular es invertible y como el parámetro estimado es $|0{,}4665| < 1$, también es estacionario. La parte estacional sigue un SMA(1) que es estacionario y como el parámetro estimado $|-0{,}8302| < 1$, también es invertible.

2. Los parámetros estimados son estadísticamente significativos.

Para comprobar si los parámetros son estadísticamente significativos o no. O también, para comprobar si se ha sobreparametrizado o infraparametrizado, se realiza un contraste de significatividad individual para cada uno de ellos. Por ejemplo, para el parámetro del AR(1) sería:

$$H_0 : \phi = 0$$
$$H_1 : \phi \neq 0$$

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 parámetros estimados de este modelo sería:

$$t = \frac{0,4675}{0,0701} = 6,669 \quad y \quad |t| = |\frac{-0,8302}{0,0860}| = 9,653$$

Como, para un nivel de significación del 5 % y los grados de libertad, el valor de la t-Student es aproximadamente 1,96, se rechaza la H_0 y, por lo tanto, son estadísticamente distintos de cero.

3. Los residuos del modelo son ruido blanco.

En la fase de validación es necesario comprobar que los residuos del modelo estimado son consistentes con el supuesto de ruido blanco. El siguiente código R, permite obtener los residuos,

```
residuos <- residuals(modelo)
ts_residuos <- data.frame(value = residuos, Time = time(residuos))
ggplot(data = ts_residuos, aes(x = Time, y = residuos)) +
  geom_line() +
  geom_line(colour = "red")
```

Figura 20.8: Gráfico de los residuos

Para contrastar que los residuos están incorrelacionados y que se comportan como un ruido blanco, se puede utilizar el estadístico de Box-Ljung. En este caso concreto, su hipótesis nula, es que los primero 40 coeficientes de correlación son cero frente a la hipótesis alternativa de que no lo sean. Para un nivel de significación del 5 %, se acepta H_0, ya que, el p-valor de la X-squared es igual a 0.5448. El código R para obtener 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
```

Además, se puede calcular ACF y PACF de los residuos para comprobar que están incorrelacionados. Los correlogramas obtenidos, con el código siguiente, muestran que 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)
```

Figura 20.9: ACF y PACF estimadas de los residuos

También se debe comprobar la capacidad de ajuste del modelo comparando los valores observados y estimados, utilizando el código R siguiente:

```
plot(ipc_ts, lwd = 2)
lines(ipc_ts - modelo$residuals, col = "red", lwd = 2)
```

Los criterios de información como el AIC se pueden utilizar para comparar entre diferentes modelos cuál se ajusta mejor a los datos. Así, será mejor el modelo que tenga un menor valor del criterio de información utilizado.

20.3.4. Predicción

Después de comprobar que el modelo es adecuado se puede utilizar para predecir. Para obtener 12 predicciones, sus correspondientes intervalos de predicción al 80 % y 95 % de confianza y su representación gráfica, con el modelo ARIMA estimado, el código R que se puede utilizar es:

Figura 20.10: Ajuste con el modelo estimado

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

RESUMEN
Los modelos univariantes de series temporales ARIMA son muy útiles para proponer el modelo que mejor capte la dinámica de una serie temporal. Fundamentalmente, se pueden utilizar para predicir y la toma de decisiones.

Figura 20.11: Predicciones con el modelo ARIMA(1,1,0)(0,1,1)12 estimado

Capítulo 21

Análisis discriminante

Mª Leticia Meseguer Santamaría

21.1. Introducción

El **análisis discriminante** es una técnica de dependencia, donde se establece la relación entre una única variable dependiente cualitativa o categórica, cuyas clases identifican a los grupos preexistentes, y un conjunto de variables independientes cuantitativas, denominadas clasificadoras o predictoras. Su **objetivo** es clasificar individuos en distintos grupos preexistentes, a partir de la información de un conjunto de variables clasificadoras, la información se sintetiza en lo que se denominan **funciones discriminantes**. Por tanto, se identifican dos finalidades: la descriptiva, caracterizar la separación entre grupos, determinando la contribución de cada variable clasificadora a la separación de los individuos; y la predictiva, dónde se clasificará un individuo nuevo.

Dependiendo de sus características se diferencian varios casos:

Dos grupos y una variable clasificadora:

Es el supuesto más simple, son dos grupos (I y II) y un conjunto de individuos para clasificar a partir de la información de una sola variable. En este caso las distribuciones en los grupos I y II solo difieren en la media, Como se muestra en la Figura 21.1.

El **punto de corte (C)** es:

$$C = \frac{\overline{x_I} + \overline{x_{II}}}{2} \tag{21.1}$$

Y el **criterio de asignación** para cada individuo:

$$Si \;\; X_i < C \in grupo \;\; I \;\; y \;\; si \;\; X_i > C \in grupo \;\; II \tag{21.2}$$

El error que podamos cometer depende del nivel de solapamiento de las dos distribuciones.

Figura 21.1: Dos grupos y una variable clasificadora

Dos grupos y p variables clasificadoras:

El objetivo es encontrar un criterio que permita separar los dos grupos. Para ello se necesita obtener la **función discriminante de Fisher**, que se plantea como una combinación lineal de las p variables clasificadoras:

$$D = u_1X_1 + u_2X_2 + ... + u_pX_p \tag{21.3}$$

La **punturación discriminante del individuo i-ésimo** es:

$$D_i = u_1X_{1i}1 + u_2X_{2i} + ... + u_pX_{pi} \tag{21.4}$$

Se trata de obtener el valor teórico como combinación lineal de las variables clasificadoras que mejor discriminen entre los grupos definidos a priori, por lo tanto, la función discriminante que maximiza las variabilidad entre-grupos respecto de la intra-grupos.

El **punto de corte (C)** se obtiene evaluando la función discriminante en cada grupo y promediando:

$$D_I = u_1X_{1I}1 + u_2X_{2I} + ... + u_pX_{pI} \tag{21.5}$$

$$D_{II} = u_1X_{1II}1 + u_2X_{2II} + ... + u_pX_{pII} \tag{21.6}$$

$$C = \frac{\overline{D_I} + \overline{D_{II}}}{2} \tag{21.7}$$

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

$$Si \;\; D_i < C \in grupo \;\; I \;\; y \;\; si \;\; D_i > C \in grupo \;\; II \tag{21.8}$$

G grupos y p variables:

Se obtinen más de una función discriminante, concretamente, el mínimo de g-1 y p.

Esta matriz no es simétrica, por lo que las funciones no son ortogonales.

El resultado proporcionará las funciones discriminantes que maximizan la variabilidad entre-grupos respecto de la intra-grupos, indicando la importancia relativa de cada una, de tal forma que la primera función será la que más discrimine, la segunda, discriminará menos y ,así, hasta la última.

21.2. Tipos de análisis discriminantes:

LDA, Análisis discriminante lineal:

Es el que se ha visto hasta ahora, las variables predictoras (p) están distribuidas normalmente y las clases tienen varianzas idénticas (si p=1) o matrices de covarianza idénticas (si p>1).

QDA, Análisis discriminante cuadrático:

Elimina el supuesto de que la matriz de covarianzas sea la misma para cada clase, por lo que aumenta su fiabilidad; ahora, se calculará para cada función discriminante k (k=1,2, 3. . . .) y no se descartan los términos cuadráticos, por lo tanto, la función discriminante tendrá términos de segundo orden.

La función discriminate viene dada por:

$$\delta_k(x) = \frac{1}{2}log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log \Pi_k \tag{21.9}$$

Donde Σ_k es la matriz de covarianzas.

La regla de clasificación es:

$$G = \underset{x}{\text{argmax}}\delta_k(x) \tag{21.10}$$

Los límites de decisión son ecuaciones cuadráticas de x.

Se recomienda QDA si el conjunto de entrenamiento es grande o en el supuesto de que una matriz de covarianzas igual para todas las variables no sea sostenible.

MDA, Análisis discriminante mixto:

Cada clase es una mezcla normal de subclases, en la que cada grupo de datos tiene una probabilidad de pertenecer a cada clase. Se asume la igualdad de la matriz de covarianza entre clases.

21.3. Procedimiento con R: las funciones `lda()`, `qda()` y `mda()`

ESTO HAY QUE REDACTARLO ENTERO

IDEA: CUANDO SE EXPLIQUE EL LINEAL, HACER EJEMPLO DEL LINEAL, Y ASÍ CON EL RESTO

A continuación, se indican los pasos a seguir en la realización del análisis y un ejemplo de cada tipo (lda, qda y mda) realizados con R

Etapas:

- **Carga de paquetes y librerías**:
  - library(tidyverse)
  - library(caret)
  - library(MASS)—> Específica para los modelos LDA y QDA
  - library(klaR)—> Específicas para los modelos QDA
  - library (mda)—> Específica para el modelo MDA
- **Preparación de datos**:

  Se utilizará y cargará la base de datos **iris**, disponible para R, que consta de 150 observaciones y 5 variales, 4 numéricas, que serán las predictoras, y una categórica, sobre la que se realiza el análisis, con tres categorías: setosa, versicolor y virginica.

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

  Se dividen los datos entre la muestra, 80 % para el entrenamiento y 20 % para el test.

  Para evitar distorsiones, es recomendable estandarizar las predictoras continuas, en R las variables categóricas se ignoran automáticamente.
- **Análisis: LDA, QDA y MDA**

  Las distribuciones univariadas deben estar distribuidas normalmente, si no fuera así, se deberán transformar utilizando rog y root (distribuciones exponenciales) y Box-Cox (distribuciones sesgadas).

  Funciones lda, qda (del paquete MASS) y mda (del paquete mda), para calcular los modelos.
- **Resultados** de lda, qda y mda:
  - **Probabilidades previas** (*Prior probabilities of groups*): Proporciones de cada grupo en el total de la muestra, en este caso, cada grupo un 33.33 %.
  - **Media de cada grupo** (*Group means*): Proporciona las medias de cada grupo y establece para cada individuo la probabilidad de pertenecer a los grupos.
  - **Coeficientes de discriminantes lineales** (*Coefficients of linear discriminants*): Son los coeficientes de cada variable predictora con los que se configuran las funciones lineales, sirven para para establecer la regla de decisión.

- **Proporción de cada discriminante** (*Proportion of trace*): El peso de cada función para discriminar.

- **Gráfica**
- **Predicciones** se uilizarán las funciones **predict**, para realizar predicciones, y **table**, para la tabla que indica la clasificación del modelo.
- **Precisión** o ajuste del modelo, mediante la función **mean** se calculará la precisión clasificadora del modelo.

Resolución con R

```
# Carga de librerías y datos
library(caret)
library(MASS)
library(klaR)
library(mda)
data("iris")

# 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, ]

# Estimar parámetros de preprocesamiento (estandarizar)
preproc_param <- entrenamiento_d |>
  preProcess(method = c("center", "scale"))

# Transformar los datos usando los parámetros estimados
entrenamiento_t <- preproc_param |> predict(entrenamiento_d)
test_t <- preproc_param |> predict(test_d)
```

MODELO LDA

```
# Estimación del modelo LDA
modelo_lda <- lda(Species ~ ., data = entrenamiento_t)
modelo_lda
#> Call:
#> lda(Species ~ ., data = entrenamiento_t)
#>
#> Prior probabilities of groups:
#>     setosa versicolor  virginica
#>  0.3333333  0.3333333  0.3333333
```

```
#>
#> Group means:
#>            Sepal.Length Sepal.Width Petal.Length Petal.Width
#> setosa       -1.0112835  0.78048647   -1.2900001  -1.2453195
#> versicolor    0.1014181 -0.68674658    0.2566029   0.1472614
#> virginica     0.9098654 -0.09373989    1.0333972   1.0980581
#>
#> Coefficients of linear discriminants:
#>                     LD1         LD2
#> Sepal.Length  0.6794973  0.04463786
#> Sepal.Width   0.6565085 -1.00330120
#> Petal.Length -3.8365047  1.44176147
#> Petal.Width  -2.2722313 -1.96516251
#>
#> Proportion of trace:
#>    LD1    LD2
#> 0.9902 0.0098
# Probabilidades a priori: 0,3333 para cada grupo
# Coeficientes y funciones discriminantes:
#  D1=0,6795*SL+0,6565*SW-3,8365*PL-2,2722*PW
#  D2=0,0446*SL-1,0033*SW+1,4418*PL-1,9651*PW
# Proporción discriminar de D1 y D2, 0,9902 y 0,0098, respectivamente

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

```
# Predicciones del modelo LDA
predicciones_lda <- modelo_lda |> predict(test_t)
table(test_t$Species, predicciones_lda$class, dnn = c("Clase real", "Clase
↪  predicha"))
#>             Clase predicha
#> Clase real   setosa versicolor virginica
#>   setosa         10          0         0
#>   versicolor      0         10         0
#>   virginica       0          1         9

# Precisión del modelo LDA
mean(predicciones_lda$class == test_t$Species)
#> [1] 0.9666667

# Ajusta al 96,9667%.
```

MODEL QDA

```
# Estimación del modelo QDA
modelo_qda <- qda(Species ~ ., data = entrenamiento_t)
modelo_qda
#> Call:
#> qda(Species ~ ., data = entrenamiento_t)
#>
#> Prior probabilities of groups:
#>     setosa versicolor  virginica
#>  0.3333333  0.3333333  0.3333333
#>
#> Group means:
#>            Sepal.Length Sepal.Width Petal.Length Petal.Width
#> setosa       -1.0112835  0.78048647   -1.2900001  -1.2453195
#> versicolor    0.1014181 -0.68674658    0.2566029   0.1472614
#> virginica     0.9098654 -0.09373989    1.0333972   1.0980581
# Probabilidades a priori: 0,3333 para cada grupo
# El matriz de las medias de los grupos para cada una de nuestras varialbles
↪  según la variable predictora.

# Gráfico Modelo QDA, se utiliza la función partimat que muestra el
↪  comportamiento de cada variable y su error.
partimat(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
↪  data = entrenamiento_t, method = "qda")
```

```
# Predicciones del modelo QDA
predicciones_qda <- modelo_qda |> predict(test_t)
table(test_t$Species, predicciones_qda$class, dnn = c("Clase real", "Clase
↪  predicha"))
#>              Clase predicha
#> Clase real    setosa versicolor virginica
#>   setosa          10          0         0
#>   versicolor       0         10         0
#>   virginica        0          1         9

# Precisión del modelo QDA
mean(predicciones_qda$class == test_t$Species)
#> [1] 0.9666667
# Ajusta al 96,9667%.
```

MODEL MDA

```
# Estimación del modelo MDA
modelo_mda <- mda(Species ~ ., data = entrenamiento_t)
modelo_mda
#> Call:
#> mda(formula = Species ~ ., data = entrenamiento_t)
#>
#> Dimension: 4
#>
#> Percent Between-Group Variance Explained:
#>     v1     v2     v3     v4
#>  93.36  97.61  99.64 100.00
#>
```

```
#> Degrees of Freedom (per dimension): 5
#>
#> Training Misclassification Error: 0.01667 ( N = 120 )
#>
#> Deviance: 9.704
# La salidas muestran los porcentajes, los grados de libertad y el error de
↪  clasificación.

# Predicciones del modelo MDA, su cálculo requiere del uso de data.frame y no
↪  se obtiene la categoría class, sino que se calcula directamente.
predicciones_mda <- data.frame(modelo_mda |> predict(test_t))

## TABLE NO DA ERRROR

# table(test_t$Species, predicciones_mda$modelo_mda, dnn=c("Clase real",
↪  "Clase predicha"))

# Precisión del modelo MDA
mean(predicciones_mda$modelo_mda == test_t$Species)
#> [1] NaN
# Ajusta al 96,9667%.
```

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 LDA (lineal), QDA (cuadrático) y MDA (mixto), que se desarrollan en R con las funciones lda, qda y mda, respectivamente.

Capítulo 22

Análisis conjunto

Mª Leticia Meseguer Santamaría

22.1. Introducción, conceptos clave y tipos de análisis

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 los preferencias del público ante el lanzamiento de nuevos productos o servicios, para adaptarlo a ellas y maximizar el éxito, evalua la sensibilidad al precio u otras características del producto y predice el comportamiento en el mercado. Mediante este análisis se pueden establecer qué atributo y nivel son los más valorados y cuantificarlos de forma relativa.

Los principales conceptos clave 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 denominadas "Partworth Utility", "Nivel 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 atributo 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 habrá unas 8 o 12.
- **Probabilidad de elección**: Probabilidad de tomar una decisión determinada considerando las utilidades parciales.
- **Modelo de comportamiento**: Modelo de decisión que se infiere desde 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 muesta de gimnasios en la que se identifican 4 atributos (características) con distintos niveles. Se realiza una encuesta sobre qué gimnasio se prefiere. Los paricipantes eligen una opción entre distintas combinaciones ofrecidas (**preguntas**). Mediante el análisis de los resultados de la encuesta se extraerá el peso de cada atributo y nivel en las respuestas (**utilidades parciales**), que describen las preferencias medias de los encuestados y se identifican los atributos y niveles más valorados y su importancia relativa.

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

Figura 22.1: Gimnasio. Atributos: Horario, Clases programadas, Limpieza y Precio

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

El análisis tradicional, de perfil completo (Full Profile) o Conjoint Value Analysis (CVA)

Se muestra al entrevistado una selección de productos ya combinados de atributos y niveles (perfiles) y se le pide que los valoren según su preferencia con una escala numérica, que será la

utilidad. Las preferencias globales de cada individuo se descomponen en valores de utilidades independientes y compatibles para cada atributo y nivel mediante métodos basados en regresión lineal múltiple, son las funciones de utilidad por 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 señalen 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 Choise-Based Conjoint (CBC)

Pretende un mayor realismo, mostrando a cada entrevistado un grupo de productos 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, que son conocidos 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 es 10, y como ventaja, que presenta un mejor sistema de elección.

22.2. Aplicación del Análisis Conjunto (etapas):

Planteamiento del problema: Se analiza la oportunidad de aplicar esta técnica dados el objetivo y los datos de la investigación.

Elección de la metodología conjoint a aplicar (dependerá del producto o servicio y de los atributos elegidos):

- CBC: hasta 6 atributos y productos habituales (de elección rápida, sin reflexión).
- ACA: Productos de elección rápida y reflexiva. Más de 10 atributos.
- CVA: Productos de elección rápida y reflexiva. Desde 6 hasta 10 atributos.

La selección de elementos: **Atributos**, se escogerán los que condicionan la elección, es decir, los que explican sus preferencias y permiten diferenciar bien entre los productos o servicios, deben ser lo más independientes entre sí; **los niveles**, determinados dentro de cada atributo, deben ser mutuamente excluyentes y cubrir todo el rango de posibilidades; y, en su caso, **perfiles y escenarios**, dependerán de la metodología a usar.

Creación de estímulos: Se utilizarán diseños factoriales fraccionados ortogonales que reducen el número de perfiles que se le muestran al entrevistado.

Forma de presentación, dependerá de la metodología elegida, se exponen ejemplos en la Fig. 22.2:

- CVA: *Matriz "Trade-off" o de comparaciones*: El entrevistado valora la combinación de atributos y niveles. Sólo es válido para dos atributos; *Perfiles completos para ordenar*: Se elaboran perfiles de cada producto o servicio utilizando sólo un nivel de cada atributo y el encuestado los valora (ordena) según sus preferencias; y *Perfil determinado para valorar*: el encuestado la valore según sus preferencias.
- ACA: *Comparaciones Pareadas*: Se comparan dos perfiles incompletos.
- CBC: *Elección de un perfil*: 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.

Estimación de las utilidades

- CVA: Se utiliza, por lo general, un modelo de Mínimos Cuadrados Ordinales (OLS), mediante el cual se determinan las utilidades parciales o parth-worth, que indican las preferencias del encuestado mediante un modelo aditivo lineal en relación a los niveles de los atributos considerados. La Utilidad total de un perfil es la suma de las parth-worth, que se podrá estimarse para cada individuo o para el total de la muestra.
    - Modelo de preferencias totales de cada perfil (i) para el total de la muestra:

$$\widehat{U_i} = \beta_0 + \sum_{j=1}^{J} \widehat{u_{jk}} \tag{22.1}$$

donde: $\widehat{U_i}$ es la utilidad global para el perfil i; $\widehat{u_{jk}}$ es la utilidad parcial asociada con el nivel k del atributo j; y β_0 esla constante de regresión.

- Modelo de preferencias totales de cada perfil (i) para cadad individuo (z):

$$\widehat{U_{iz}} = \beta_{0z} + \sum_{j=1}^{J} \widehat{u_{jkz}} \tag{22.2}$$

donde: $\widehat{U_{iz}}$ es la utilidad total del perfil i para el perfil z; $\widehat{u_{jkz}}$ es la utilidad parcial asociada con el nivel k del atributo j para cada individuo z; y β_{0z} esla constante de regresión para cada individuo z.

Interpretación de resultados, se estudian:

Figura 22.2: Formas de presentación

- La importancia de los atributos y niveles.
- Las utilidades globales, tanto de los productos de forma individual como en comparación con otros.
- La elasticidad de alguna determinada variable, normalmente el precio, en relación a la elección que hace el sujeto.

22.3. Procedimiento con R: la función `Conjoint()`

Se hará uso de la librería **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).

Ejemplo: Se desea conocer el porqué unos tés son preferidos a otros, para ello se seleccionan 4 atributos, con sus correspondiente niveles:

- 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.

Se trata de un análisis tradicional con perfiles incompletos.

Se cargan la librería y los datos:

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

Se muestran los datos, se puede utilizar el comando *str()* para que solo muestre unos pocos.

```
tlevn
tprof
str(tpref)
str(tprefm)
tsimp
```

Con la función *conjoint* para el *total de la muestra*, devolverá las utilidades de los niveles (partworths), el vector de porcentajes (la suma debe ser 100) de la importancia de los atributos y las gráficas correspondientes.

OJO: ES INTERACTIVA Y ME DA ERROR, SACAR LOS RESULTADOS MÁS IMPORTNATES Y LOS PLOTS A COLOR Y PONERLOS COMO IMÁGEN

```
library(conjoint)
Conjoint(y = tpref, x = tprof, z = tlevn)
```

Interpretación del resultado:

- Los coeficientes del modelo de regresión referidos a todos los niveles, en el que se señala con asteriscos los que resultan significativos, así como su grado de ajuste.
- Ajuste del modelo de regresión.
- Las utilidades por niveles o part-worth.
- La importancia de los atributos o factores, en este ejemplo el atributo con más peso es la variedad (32.22); el siguiente es la presentación (27.15); después; el siguiente, el precio (24.76); y por último, el aroma (15.88). Esta información también se nos muestra de forma gráfica y agrupada.

Además, analizando los datos por niveles (parth-worth) se puede señalar que:

- El atributo más importante: Variedad, con una utilidad parcial de 32.22, y la variedad de té negro (black) es el nivel más importante, con 0.6149.
- El segundo atributo más importante: Presentación, con una utilidad parcial de 27.15, y en hojas (leafy) es el nivel más importante, con 0.7529.
- El tercer atributo más importante: Precio, con una utilidad parcial de 24.76, y el precio bajo (low) es el nivel más importante, con 0.2402.
- El cuarto atributo más importante: Aroma, con una utilidad parcial de 15.88 y que sea aromático (yes) es el nivel más importante, con 0.4108.

En la siguiente tabla se exponen otras funciones en R para determinados cálculos:

Función en R	Utilidad-resultado
caUtilities (y=tprefm , x=tprof, z=tlevn)	Utilidades para cada nivel sobre el total de los individuos
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 para los encuestados individualmente
colMeans(caTotalUtilities(y=tprefm , x=tprof))	Utilidades totales por perfil
ShowAllUtilities (y=tprefm , x=tprof, z=tlevn)	Todas las utilidades

Función en R	Utilidad-resultado
caImportance (y=tprefm , x=tprof)	Importancia relativa de cada atributo considerando el total de individuos
print(caImportance (y=tprefm , x=tprof)	Para obtener la gráfica
caModel (y=tprefm[20,], x=tprof)	Preferencias para un solo individuo, por ejemplo el 20.
caUtilities (y=tprefm[20,] , x=tprof, z=tlevn)	Utilidades de los niveles para un individuo (p.e. 20)

Aplicación: Segmentación

- Objetivo: identificar grupos de encuestados con preferencias similares.
- Función a utilizar: caSegmentatión, devuelve un análisis de conglomerados dividiendo a los individuos en n clusters usando el método de k-means. Se tomarán 3 clusters.

El vector de agrupación que se mostrará contiene los valores del cluster.

```
caSegmentation(y = tprefm, x = tprof, c = 3)
```

Para la gráfica se utilizará la función plotcluster

```
segments <- caSegmentation(y = tprefm, x = tprof, c = 3)
summary(segments)
require(fpc)
plotcluster(segments$util, segments$sclu)
```

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 la sensibilidad de alguna característica, por ejemplo el precio, a los cambios. Función principal de uso en R: **conjoint**.

Capítulo 23

Análisis de tablas de contingencia

José-María Montero

23.1. Introducción

23.1.1. Motivación

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 (poco más), y en la esfera de la Inferencia Estadística 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?

23.1.2. 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 $\{A_1, A_2, \ldots, A_R\}$ y $\{B_1, B_2, \ldots, B_C\}$ 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	$\ldots$	Nivel B_C	**Total**
	Nivel A_1	n_{11}	n_{12}	$\ldots$	n_{1C}	$n_{1.}$
	Nivel A_2	n_{21}	n_{22}	$\ldots$	n_{1C}	$n_{2.}$
	.	.	.	.	.	.
Factor A	.	.	.	.	.	.
	.	.	.	.	.	.
	Nivel A_R	n_{R1}	n_{R2}	$\ldots$	n_{RC}	$n_{R.}$
	Total	$n_{.1}$	$n_{.2}$	$\ldots$	$n_{.C}$	N

Ejemplo 23.1 (Tabla de contingencia con dos factores y dos niveles cada factor)**.** Se toma una muestra de 80 ayuntamientos de una CC.AA. y se anota el signo político del equipo gubernamental y si prestan o no públicamente el servicio X.

		Prestación	pública	
		NO	SI	**Total**
Signo	Avanzados	14	28	**42**
político	Ilustrados	6	32	**38**
	Total	**20**	**60**	**80**

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 no?, ¿la asociación entre los factores "Signo político del equipo gubernamental" y "Prestación pública del servicio X" es muy intensa? Del análisis de la tabla de contingencia anterior nos ocuparemos posteriormente. Por el momento, nos centramos en su construcción a partir de la base de datos de este ejemplo y en el cálculo de porcentajes (en este caso, por fila). Se procede como sigue:

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

ayuntam %$%
  summarytools::ctable(`signo_gob`, `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%)
#> ------------ ------ ------------ ------------ -------------
```

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.

23.1.3. 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 siguientes[1]:

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

Ejemplo 23.2. 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

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

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	
		NO	SI	**Total**
Tipo de	A	N_{11}	N_{12}	**22**
judía	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.

- **Tipo 2: Solo se fijan los totales marginales de uno de los factores**

Ejemplo 23.3. 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 23.4. 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 densiometría de columna y rellenó 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**

23.2. Contraste de independencia en tablas 2×2

23.2.1. Introducción

Como se avanzó en la sección 23.1.1, 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, nos centraremos inicialmente en 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.

23.2.2. Planteamiento general del contraste exacto de independencia

- Hipótesis: H_0: Los factores son independientes. H_1: Están asociados.[2]

- 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 superior 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.

23.2.3. Algoritmo para la realización del contraste exacto de independencia

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

1. Selección de la 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

[2] También 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.

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 sección 23.2.5 se comentan algunas cuestiones de interés cuando el diseño de muestreo es de tipo 2 o tipo 3.

23.2.4. Contraste de independencia: Diseño Tipo 1

23.2.4.1. Contraste exacto (test exacto de Fisher)

Considérese el ejemplo 23.2[3]. 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))
```

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

Según el algoritmo expuesto en la sección 23.2.3, 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: $n_{ij} = \frac{n_{i.}n_{.j}}{N}, \forall i, j$, donde $\frac{n_{i.}n_{.j}}{N} = \hat{E}_{ij}$ 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,

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

[4] El 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, las tablas seleccionadas serían, en el primer caso, todas menos la T_7, y en el segundo T_0 y T_1.

en valor absoluto, y tomando de referencia, por ejemplo, la celda {1,1}: $\hat{D}_{11} = n_{11} - \hat{E}_{11} = n_{11} - \frac{n_{1.}n_{.1}}{N} \geq \hat{D}_{11obs}$.

En nuestro caso, 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):

$\boldsymbol{T}_0$: -3,85; $\boldsymbol{T}_1$: -2,85; T_2: -1,85; T_3: -0,85; T_4: 0,15; T_5: 1,15; T_6: 2,15; $\boldsymbol{T_7}$: 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 $\left|\frac{n_{11}}{n_{1.}} - \frac{n_{21}}{n_{2.}}\right| = |\hat{D}_{11}|\frac{N}{n_{1.}n_{2.}}$.

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: $\binom{n}{n_{1.}}\binom{n}{n_{.1}} = \frac{n!}{(n_{1.}!n_{2.}!)}\frac{n!}{(n_{.1}!n_{.2}}$.

Por tanto, el cociente entre ambas es: $P = \frac{n_{1.}!n_{2.}!n_{.1}!n_{.2}!}{n!n_{11}!n_{12}!n_{21}!n_{22}!}$.

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

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:

```
# 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 sección 23.4, dedicada a las medidas de asociación en tablas 2x2, se cuantificará su intensidad.

23.2.4.2. Contraste aproximado

El contraste exacto de independencia se basa en el cálculo de la probabilidad de ocurrencia de cada una de las tablas que se alejen de la hipótesis de independencia tanto o más que la observada y su comparación con α. Cuando el procedimiento de muestreo es de tipo 1, si se toma como celda de referencia, por ejemplo, la {1,1} la probabilidad a comparar con α es: $P\left(N_{11} - \frac{n_{1.}n_{.1}}{n}\right)^2 \geq \left(n_{11} - \frac{n_{1.}n_{.1}}{n}\right)^2$.

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 $E(N_{ij}) = \frac{N_{ij}}{n_{i.}n_{.j}}$ 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^2_{ajd}) y es tal que $\chi^2_{ajd} = \frac{n-1}{n}\frac{n(n_{11}n_{22}-n_{21}n_{12})^2}{n_{1.}n_{.1}n_{2.}n_{.2}}$, 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.

```
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 $\chi^2_{ajd} = 5,54073$ y $P\left(\chi^2_{ajd} \geq 5,54073\right) = 0,0186$.[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.

23.2.4.3. Contraste aproximado con corrección de continuidad

Como se vio en la sección 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^2_1 (que es continua), se comete un "error de continuización". Dicho error se intenta corregir incluyendo en el contraste una "corrección 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:

$$P\left(\chi^2_1 \geq \frac{(n-1)(|n_{11}n_{22} - n_{21}n_{12}| - 0,5n)^2}{n_{1.}n_{.1}n_{2.}n_{.2}}\right) \leq \alpha,$$

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 ($\chi^2_{ajd,CCY}$).

En el ejemplo propuesto,

```
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
```

[5] Este "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.

$\chi^2_{CCY} = 3,8337$ y entonces $\chi^2_{ajd,CCY} = \frac{39}{40} \times 3,8337 = 3,7636$; y como $P\left(\chi^2_1 \geq 3,7636\right) = 0,0524$, 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^2_{CCY} y $\chi^2_{ajd,CCY}$; 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 multiplos 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).

23.2.5. 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

$$P\left(N_{11} = n_{11}; N_{12} = n_{12}; N_{21} = n_{21}; N_{22} = n_{22} | N = n\right) = \binom{n_{1.}}{N_{11}}\binom{n_{2.}}{N_{21}}\left(\frac{N_{.1}}{n}\right)^{N_{.1}}\left(\frac{N_{.2}}{n}\right)^{N_{.2}} \tag{23.1}$$

El estadístico de contraste en el test aproximado viene dado por $\chi^2 = \frac{n(n_{11}n_{22}-n_{21}n_{12})^2}{n_{1.}n_{.1}n_{2.}n_{.2}}$, y por $\chi^2_{CC} = \frac{n\left(|n_{11}n_{22}-n_{21}n_{12}|-\frac{f}{2}\right)^2}{n_{1.}n_{.1}n_{2.}n_{.2}}$ 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:

$$\begin{gathered} P\left(N_{11} = n_{11}; N_{12} = n_{12}; N_{21} = n_{21}; N_{22} = n_{22} | N = n\right) = \\ \frac{n!}{n_{1.}!n_{2.}!n_{.1}!n_{.2}!}\left(\frac{n_{1.}}{n}\frac{n_{.1}}{n}\right)^{n_{11}}\left(\frac{n_{1.}}{n}\frac{n_{.2}}{n}\right)^{n_{12}}\left(\frac{n_{2.}}{n}\frac{n_{.1}}{n}\right)^{n_{21}}\left(\frac{n_{1.}}{n}\frac{n_{.2}}{n}\right)^{n_{22}} \end{gathered} \tag{23.2}$$

El test aproximado, en este caso, es un test razón de verosimilitudes donde el estadístico de contraste, $G = -2ln\frac{n_{1.}^{n_{1.}}n_{2.}^{n_{2.}}n_{.1}^{n_{.1}}n_{.2}^{n_{.2}}}{n_{11}^{n_{11}}n_{21}^{n_{21}}n_{12}^{n_{12}}n_{22}^{n_{22}}n^n}$, 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. Mi recomendación es no usar ninguna, 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 sección 23.3.2.

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

23.3.1. Introducción

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 coherencia[7]. 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.

23.3.2. 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:

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

que, si no se incumple el requisito expuesto en 23.3.1, 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 $\left(N_{ij} - \hat{E}_{ij}\right)^2$ 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.

[6] La 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 simetrice 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.

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

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 al 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, $G = -2ln\Lambda$, 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 presidente 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:

```
data("ayuso")
tabla_ayuso <- table(ayuso)
tabla_ayuso
#>                opinion
#> zona            n1_nefasta 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_nefasta  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 %).

23.3.3. 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.

23.4. Medidas de asociación en tablas 2×2

23.4.1. Introducción

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 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 2x2 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 derivados de la dirección de la misma (positiva y negativa) ya se se definieron en 23.2.2. Por lo que se refiere a los límites de la intensidad de la misma, 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 2 [8]: - 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.

[8] La 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.

23.4.2. La $\hat{Q}$ de Yule

En caso de independencia, las frecuencias observadas coinciden con las estimaciones de la 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 $\hat{D}_{11}$, $\left[-\frac{n_{12}n_{21}}{n}; \frac{n_{12}n_{21}}{n}\right]$ 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:

- 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.

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

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

23.4.3. Otras medidas de asociación para tablas 2×2

23.4.3.1. 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 el 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".

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

23.4.3.2. Odds ratio o cociente de posibilidades[9]

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

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 mayor en el nivel 1 del factor A que en 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 menor 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.

[9] Para no confundirlo con el riesgo relativo.

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.

```
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
```

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

23.5.1. Introducción

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.

23.5.2. Medidas derivadas del estadístico Chi-cuadrado

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 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 $V = T$.

```
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,

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

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 %).

23.5.3. 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 que 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 $P(i) = N - \text{máx}_j\, n_{.j}$ y $P(ii) = N - \sum_{j=1}^{C} \text{máx}_j\, n_{.j}$. En el caso en que A sea el factor a explicar: $\lambda = \frac{P(i)-P(ii)}{P(i)}$, con $P(i) = N - \text{máx}_i\, n_{i.}$ y $P(ii) = N - \sum_{i=1}^{R} \text{máx}_i\, n_{i.}$ En caso de no tener claro cual es el factor a explicar, se utiliza la media agregativa de las dos medidas anteriores $\lambda = \frac{\sum_{j=1}^{C} \text{máx}_i\, n_{i.} - max_i n_{i.} + \sum_{i=1}^{R} \text{máx}_j\, n_{.j} - \text{máx}_j\, n_{.j}}{2N - \text{máx}_i\, n_{i.}\, \text{máx}_j\, n_{.j}}$. 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) es que exige determinar el factor explicativo y el factor a explicar. Otra es su sensibilidad a totales marginales desequilibrados; en este caso toma valores anormalmente bajos.

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

Nuestro ejemplo práctico es un caso 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.

23.5.4. 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 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: $\hat{R}_{ij} = n_{ij} - \hat{E}_{ij}$; $\hat{R}_{ij}(est) = \frac{n_{ij}-\hat{E}_{ij}}{\sqrt{\hat{E}_{ij}}}$; $\hat{R}_{ij}(est;ajd) = \frac{\hat{R}_{ij}(est)}{\sqrt{(1-\frac{n_{i.}}{N})(1-\frac{n_{.j}}{N})}}$

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

```
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
```

En el ejemplo que nos ocupa, asumiendo $\alpha = 0,05$, las fuentes de asociación son "Madrid municipio-excelente" a costa de "buena", y "Madrid metropolitano-buena" a costa de "excelente".

23.6. Contrastes de independencia en tablas multidimensionales

En tablas con más de dos factores (nos centramos en 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 $\chi^2 = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{m=1}^{M}\frac{\left(n_{ijm}-\hat{E}_{ijm}\right)^2}{\hat{E}_{ijm})}$, con los siguientes grados de libertad (g.l.) y $\hat{E}_{ijm}$ bajo la correspondiente H_0:

Independencia global:

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

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 $\hat{E}_{ijm} = \frac{n_{ij.}n_{..m}}{N}$

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 $\hat{E}_{ijm} = \frac{n_{i.m}n_{.j.}}{N}$

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 $\hat{E}_{ijm} = \frac{n_{.jm}n_{i..}}{N}$

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 $\hat{E}_{ijm} = \frac{n_{i.m}n_{.jm}}{N}$

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 $\hat{E}_{ij.} = \frac{n_{i.m}n_{.jm}}{N}$

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 se proporciona una selección de medidas de asociación. Finalmente, se hace una breve incursión en el ámbito de las tablas multidimensionales.

Parte V

Machine learning supervisado

Capítulo 24

Árboles de clasificación y regresión

Ramón A. Carrasco e Itzcóatl Bueno[1]

24.1. Introducción

Los árboles de decisión, donde decisión es el término genérico para clasificación y regresión, son modelos que se utilizan principalmente para la resolución de problemas de clasficación, aunque también son aplicables a la predicción de valores numéricos, esto es, como modelos de regresión. De ahi, que sean conocidos como árboles de clasificación y regresión (CART). Los árboles de decisión predicen categorías y variables cuantitativas utilizando variables de entrada númericas y categóricas. Algunos ejemplos de arboles de decisión para clasificación y regresión son:

- **Clasificación**: Tomar la decisión de que empleados promocionar en base a sus méritos, capacidades, edad, etc. O por ejemplo, organizar un partido de ténis en base a la climatología prevista. Este ejemplo se muestra gráficamente en la figura 24.1. En este ejemplo, en base a los registros climatológicos de los partidos que ya se hayan jugado el algoritmo podrá tomar decisiones. Así, si un nuevo día se quiere jugar al tenis se deberán aportar los datos de previsión, fuerza del viento y fuerza de la humedad. En caso de ser un día nublado, el algoritmo nos sugerirá que juguemos. En caso de ser soleado, comprobará el nivel de humedad y en caso de ser debíl recomendará que se juegue el partido. Lo mismo pasará si la previsión es de lluvia pero comprobando la fuerza del viento.

- **Regresión**: Siguiendo con el ejemplo del partido de tenis, también se puede aplicar un arbol de decisión para determinar cuantas horas jugar de acuerdo a la climatología. En

[1]Universidad Complutense de Madrid

Figura 24.1: Ejemplo de árbol de decisión.

este ejemplo, en la figura 24.1 se sustituirían la predicción dicotómica SI/NO por un valor númerico. Por ejemplo, el algoritmo puede sugerir jugar 5 horas al tenis si el día esta soleado pero tiene una humedad débil, y 3.5 horas si está soleado pero la humedad es fuerte. También puede decidir que si la previsión es nublada se jueguen 4 horas.O en caso de lluvia podría decidir que el partido con viento fuerte no dure más de 0.75 horas y en caso de viento débil poder jugar 1.15 horas.

CART es un término genérico para describir algoritmos de árbol y también un nombre específico para el algoritmo original de Breiman et al. (1984) para construir árboles de clasificación y regresión. Sin embargo, existen Sin embargo, existen otros como el ID3, que se presenta en secciones posteriores, o el C4.5, que esta basado en ID3. En la tabla **??** se muestra una pequeña comparativa de estos tres algoritmos:

Tabla 24.1: Características de los principales algoritmos de árboles de decisión.

Algoritmo	Criterio de división	Tipo de variables	Estrategia de poda
ID3	Ganancia de información	Solo categóricas	No poda
CART	Indice 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

En resumen, los arboles de decisión tienen multiples ventajas. Entre las que destacan:

- Es facil 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 es valido tanto para problemas de clasificación como de regresión, y se pueden incluir tanto variables continuas como categóricas. Además, el algoritmo de árboles de decisión no requiere la estandarización de variables puesto que se basa en reglas y no en el cálculo de distancias entre observaciones.
- Trata de forma más adecuada la falta de linealidad entre las variables independientes que otros algoritmos.
- Respecto a los datos, el algoritmo de árbol de decisión hace un tratamiento automático de valores ausentes y no se resiente con las observaciones atípicas.

Sin embargo, el algoritmo tambien tiene ciertas desventajas:

- El árbol de decisión es inestable, esto es, que incluir una nueva observación de entrenamiento obliga a reconstruirlo pudiendo modificar todos los nodos en el proceso de cálculo.
- El árbol de decisión no es adecuado cuando el conjunto de datos es muy grande puesto que el modelo entrenado puede ser excesivamente complejo y provocar sobreajuste. El sobreajuste es el principal problema de los árboles de decisión, profundiza mucho en el modelo hasta convertirlo en demasiado complejo como para generalizar; lo que lleva a ajustar muy bien los datos observados y cometer muchos errores en la predicción de nuevas observaciones.
- El sobreajuste produce una gran varianza en las predicciones llevando de nuevo a muchos fallos en las predicciones y reduciendo su rendimiento.

24.2. Aprendizaje con árboles de decisión

Formalmente, un árbol de decisión es un gráfico acíclico que se inicia en un **nodo raiz**, el cual se divide en **ramas**, también conocidas como **aristas**. Después, estas ramas se unen a las **hojas**, también denominadas **nodos**, las cuales determinan **puntos de decisión**. En el ejemplo de la figura 24.1 el nodo raiz es la caja *Previsión*. Las ramas o aristas, son los tres posibles valores que puede tomar la previsión: *Soleado*, *Nublado* o *Lluvia*. Cada una de estas ramas conecta con una nueva hoja o nodo, a *Humedad* o *Viento* en los casos de soleado o lluvia. Sin embargo, en ese ejemplo, el *Nublado* representa un nodo terminal, puesto que llegado a ese punto la salida que proporcionaría el árbol es *"No jugar al tenis"*. Este proceso se repite utilizando el conjunto de datos disponible en cada hoja. Así, se genera una clasificación final cuando una hoja ya no produce ramas nuevas, y es lo que se conoce como un nodo terminal. El objetivo al construir un árbol, es mantenerlo lo más pequeño posible. Esto se consigue eligiendo una variable que divida de forma óptima los datos en conjuntos homogeneos, de tal forma que se prediga mejor la clase objetivo.

24.3. ¿Cómo se va dividiendo el árbol?

Como ya se ha mencionado, un árbol de decisión se va diviendo en dos nuevas ramas de forma recursiva, es decir, cada división esta condicionada a 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 arbol se minimice. En problemas de regresión, la función a minimizar es la suma residual de cuadrados (SSR, por las siglas en inglés de: *sum of squares due to regression*) total. Para problemas de clasificación, la partición generalmente se realiza para maximizar la reducción en la entropía.

El algoritmo CART utiliza la impureza de Gini para generar las particiones, mientras que los algoritmo ID3 y C4.5 utilizan la entropía o la ganancia de información, la cual esta relacionada con la entropía. Y como ya se ha mencionado, en regresión la función a minimizar es la suma residual de cuadrados.

24.3.1. 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 de los datos se asignaría a la clase erronea si se etiquetase al azar de acuerdo con la distribución de las clases en el conjunto de datos. Entonces, 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 puede definir como:

$$Gini(X) = 1 - \sum_{i=1}^{\kappa} p_i^2 \tag{24.1}$$

En el ejemplo de la figura 24.1 podría pasar que sucediese la siguiente situación:

Tabla 24.2: Ejemplo impureza de Gini

	$p_{SÍ}$	p_{NO}	Impureza de Gini
Soleado	0,75	0,25	0,3750
Nublado	1,00	0,00	0,0000
Lluvia	0,50	0,50	0,5000

Se puede ver que la impureza es mayor en el nodo en el que la distribución de las observaciones es uniforme entre clases, mientras que el menor valor se obtiene en el nodo en el que todas las observaciones pertenecen a una clase. Se selecciona un atributo con la menor impureza de Gini para dividir el nodo. 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 de Gini se puede definir como:

$$Gini_{\varphi}(X) = \frac{n_1}{n} Gini(X_1) + \frac{n_2}{n} Gini(X_2) \tag{24.2}$$

Al entrenar un árbol de decisión, se elige para dividir el nodo la variable que proporciona el menor $Gini_{\varphi}(X)$. Para conocer la ganancia de información para una variable, las impurezas ponderadas de las ramas se restan de la impureza original. La ganancia de Gini se calcula tal que:

$$\Delta Gini(\varphi) = Gini(X) - Gini_{\varphi}(X) \tag{24.3}$$

Siguiendo el ejemplo del árbol de decisión para saber si se puede jugar al tenis o no se tendría que obtener la impureza de gini para el nodo *niebla* o el nodo *viento*. En caso, de que para el nodo *niebla* se tuviese $P(S\text{Í}) = 0,75$ y $P(NO) = 0,25$, siendo su impureza igual a 0,375; y para el nodo *viento* $P(S\text{Í}) = 0{,}9$ y $P(NO) = 0,1$ teniendo un valor de 0,18 para su impureza de Gini. Entonces, la ganancia de Gini para cada nodo será:

$$\Delta Gini(Niebla) = 0{,}375 - 0{,}375 = 0$$
$$\Delta Gini(Viento) = 0{,}5 - 0{,}18 = 0{,}32$$

Y por tanto, el árbol tomaría como nodo a dividir el que indica que decisión tomar en caso de que la humedad sea débil o fuerte.

24.3.2. Entropía

La entropia es un concepto matemático que explica la varianza en los datos entre diferentes clases. Busca que los datos sean más homogeneos en cada capa respecto de la anterior partición. A partir de la ecuación 24.4 se calcula, expresada en bits entre 0 y 1, la entropía para cada división de variable potencial:

$$E = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \tag{24.4}$$

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 el caso de la entropía, se utiliza el logaritmo base 2 que da la unidad de bits o shannons. En cambio, el logaritmo neperiano (con base e) medirá en unidades naturales, y el logaritmo en base 10 utiliza como unidades los dits, bans o hartleys.

24.3.3. Ganancia de información

Se puede definir la ganancia de información como cuanta información proporciona una variable sobre una clase. Obtener esta información ayuda a ordenar los atributos en los nodos del árbol de decisión. La ganancia de información se obtiene tal que:

$$IG = E_{\varkappa} - E_{\varkappa+1} \tag{24.5}$$

donde $E_{\varkappa}$ representa la entropía en el nodo padre, mientras que $E_{\varkappa+1}$ es la entropia en el nodo que resulta de dividir el nodo padre. Cuanto más entropía se elimine, mayor será la ganancia de información, y por tanto, cuanto mayor sea la ganancia de información, mejor será la división.

24.3.4. Suma residual de cuadrados mínima

En el caso de un problema de regresión, el árbol elige la variable con la que seguir diviendo nodos aquella que reduce la suma de residuos al cuadrado. Esto es, que al dividir los datos utilizando la variable X en dos regiones R_1 y R_2 el error global entre la variable respuesta observada Y y el valor constante predicho por el árbol κ_i se minimice. Entonces, la función a minimizar es:

$$SSR = \sum_{i \in R_1} (y_i - \kappa_1)^2 + \sum_{i \in R_2} (y_i - \kappa_2)^2 \tag{24.6}$$

24.4. Sobreajuste

Ya se ha comentado en la sección 24.1 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. Basandose en los datos de entrenamiento, un arbol de decisión puede extraer los patrones presentes en éste y ser muy preciso en el ajuste de estos datos. Sin embargo, puede ocurrir que el árbol resultante no sea capaz de clasificar correctamente ni el conjunto de validación ni nuevas observaciones. Esto puede darse 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 realizo correctamente y no era representativa del conjunto completo. La forma de evitar el sobreajuste es controlar el crecimiento del árbol para evitar que se vuelva muy complejo.

24.5. ¿Cuánto debe crecer un árbol?

Una vez se determina la variable optima para realizar la división de los datos, se generan dos nuevas hojas y como se ha mencionado el proceso se repite recursivamente. Entonces, ¿cuándo se detiene? se debe determinar un criterio de parada adecuado. Por ejemplo, se puede determinar una profundidad máxima de árbol para que no se vuelva excesivamente complejo. Sin embargo, esto genera una nueva pregunta ¿qué profundidad máxima determinar? Si el árbol tiene mucha profundidad se hará muy complejo y sobreajustará los datos de entrenamiento resultando en un ajuste deficiente a nuevas observaciones.

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.

24.5.1. La parada temprana

La parada temprana restringe el crecimiento del árbol de forma explicita. 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 un extremo, si se permite que un nodo terminal solo contuviese una observación esta actuaría como predicción. De este modo, los resultados 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 en consecuencia se reduce la varianza.

24.5.2. La poda

El otro enfoque es el de la poda. Éste consiste en dejar crecer un árbol muy profundo y complejo y despues podarlo para encontrar un sub-árbol óptimo. El sub-árbol optimo se encuentra utilizando un parametro de complejidad del coste (ζ) que penaliza la función objetivo de la partición por el número de nodos terminales del árbol (τ) tal que:

$$\text{mín}\{SSR + \zeta|\tau|\} \tag{24.7}$$

Para un valor dado de ζ se encuentra el árbol podado más pequeño que tenga el error penalizado más bajo. Las penalizaciones más bajas tienden a producir modelos más complejos, y en consecuencia árboles más grandes. Por otro lado, las mayores penalizaciones dan como resultado árboles pequeños. En conclusión, a medida que un árbol crece, el SSR debe tener una reducción mayor que la penalización por la complejidad en los costes.

24.6. El algoritmo ID3 para la construcción de un árbol de decisión

Dado un conjunto de observaciones pertenecientes a dos clases {0,1}, se quiere construir un árbol de decisión que permita predecir a que clase pertenece una observación dado un vector de características. Existen múltiples formulaciones del algoritmo de árboles de decisión. En este libro se presenta el *ID3*. En este caso, el criterio de optimización es la verosimilitud logarítmica media:

$$\frac{1}{N}\sum_{i=1}^{N} y_i \ln f_{ID3}(x_i) + (1-y_i)\ln(1-f_{ID3}(x_i)) \tag{24.8}$$

donde f_{ID3} es un árbol de decisión. El algoritmo ID3 obtiene la solución óptima construyendo un modelo no parametrico $f_{ID3}(x) = P(y = 1|x)$. El funcionamiento del algorítmo (Quinlan (1983)) es el siguiente.

Sea $\mathbb{S}$ el conjunto de observaciones etiquetadas. En el nodo raiz del árbol se incluyen todas las observaciones $\mathbb{S} = \{(x_i, y_i)\}_{i=1}^{N}$, empezando así con un modelo constante:

$$f_{ID3}^{\mathbb{S}} = \frac{1}{|\mathbb{S}|} \sum_{(x,y)\in\mathbb{S}} y. \tag{24.9}$$

Este modelo produce la misma predicción independientemente de los valores de entrada. Teniendo en cuenta las variables $j = 1, \ldots, p$ y los todos los umbrales t se divide el conjunto $\mathbb{S}$ en dos subconjuntos $\mathbb{S}_ = \{(x, y)|(x, y) \in \mathbb{S}, x^{(j)} < t\}$ y $\mathbb{S}_+ = \{(x, y)|(x, y) \in \mathbb{S}, x^{(j)} \geq t\}$. Estos nuevos conjuntos llevan al árbol a producir hojas nuevas, por lo que se evalua que pareja (j, t) es la mejor utilizando la ya mencionada entropía. Este proceso continua recursivamente hasta llegar a los nodos terminales.

24.7. Procedimiento con R: la funcion `rpart`

En el paquete `rpart` de **R** se encuentra la función `rpart` que se utiliza para entrenar un arbol de decisión:

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

- 'formula': Que refleja el predictor lineal $Y \sim X_1 + ... + X_p$.
- 'data': Un conjunto de datos con el que interpretar el nombre de las variables incluidas en la formula.

24.8. Aplicaciones de los árboles de decisión

En esta sección se incluyen ejemplos de distintos modelos de arboles de decisión que se pueden obtener con **R**. Usando los datos `dp_entr` incluidos en el paquete `CDR` consistentes en la información de compras de distintos productos por parte de los clientes, se pretende predecir si estos compraran o no un nuevo producto (tensiómetro digital) no incluido en el modelo. A continuación se expone el caso a resolver en este y en futuros capítulos.

24.8.1. El caso de negocio

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 lo define como el proceso de compra, venta o intercambio de bienes, servicios e información a través de redes de comunicación, comunmente Internet. La clasificación más básica del CE se hace en base al tipo de entes que se relacionan: empresas (businesses, B), consumidores (consummers, 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 consumidores finales.

En nuestro caso, la empresa "Beauty eSheep" se puede considerar 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 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 a muy buen precio, 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 ofrecerselo 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 la siguiente:

- **dp_ENTR**: incluida en el paquete `CDR`

A continuación se describe los campos de dicha tabla:

Tabla 24.3: Descripción de las variables del conjunto de datos utilizado en los ejemplos.

COLUMNA	*TIPO*	*DESCRIPCIÓN*
ind_pro11	Factor	Indicador si el cliente es consumidor del producto "Fragancia Luxury" ('S') o no ('N')
ind_pro12	Factor	Indicador si el cliente es consumidor del producto "Depiladora Eléctrica" ('S') o no ('N')
ind_pro14	Factor	Indicador si el cliente es consumidor del producto "Crema Luxury" ('S') o no ('N')
ind_pro15	Factor	Indicador si el cliente es consumidor del producto "Smartwatch Fitness" ('S') o no ('N')
ind_pro16	Factor	Indicador si el cliente es consumidor del producto "Kit Pesas Inteligentes" ('S') o no ('N')
ind_pro17	Factor	Indicador si el cliente es consumidor del producto "Estimulador Muscular" ('S') o no ('N')

COLUMNA	*TIPO*	*DESCRIPCIÓN*
importe_pro11	Doble	Importe neto global consumido por el cliente en ese producto en euros
importe_pro12	Doble	Importe neto global consumido por el cliente en ese producto en euros
importe_pro14	Doble	Importe neto global consumido por el cliente en ese producto en euros
importe_pro15	Doble	Importe neto global consumido por el cliente en ese producto en euros
importe_pro16	Doble	Importe neto global consumido por el cliente en ese producto en euros
importe_pro17	Doble	Importe neto global consumido 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
ingresos_ano	Doble	Ingresos anuales del cliente en euros
des_nivel_edu	Factor	Descripción del nivel de educación del cliente
CLS_PRO_pro13	Factor	Clase objetivo, es un indicador si el cliente es consumidor de ese producto "Tensiómetro Digital" ('S') o no ('N')

24.8.2. Árbol de clasificación para determinar la intención de compra

Se contruye un árbol de clasificación usando todo el conjunto de entrenamiento completo sin transformar (en su escala orginal) mediante el algoritmo de Partición Recursiva y Àrboles de Regresión (Recursive Partitioning and Regression Trees, RPART) que se puede usar tanto para regresión como para clasificación.

```
library(CDR)

data(dp_entr)

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

En primer lugar, se ha cargado la librería necesaria para entrenar el modelo y los datos de

compras de los clientes para predecir si compraran o no el nuevo producto. Además, se define el método de remuestreo para el entrenamiento del modelo. El método definido es validación cruzada con 10 folds, visto en el capítulo **??**. A continuación, se determina la semilla aleatoria para hacer los resultados comparables y a su vez se entrena el modelo.

```
# se fija una semilla común a todos los modelos
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_pro13 ~ .,
  data = dp_entr,
  method = "rpart",
  metric = "ROC",
  trControl = trControl
)
```

```
ggplot(melt(model$resample[, -4]), aes(x = variable, y = value, fill =
↪  variable)) +
  geom_boxplot()
```

Figura 24.2: Resultados del modelo durante la validación cruzada.

Los resultados de validación cruzada quedan recogidos en los boxplot, por lo que podemos ver en que rangos oscilan las principales medidas, definidas **??**, de precisión del modelo (ROC, sensibilidad y especificidad). A continuación se muestra el árbol generado. Se puede observar que este árbol es muy sencillo, y por tanto es facil seguir su interpretación. En primer lugar, decide que si un cliente compra el *smartchwatch fitness* comprará el nuevo producto. En caso de no comprar el *smartchwatch fitness*, pero si comprar la *depiladora eléctrica* si comprará el

tensiómetro digital. Mientras que si no compra ninguno de esos dos productos no comprará el nuevo producto.

```
# se pinta el árbol obtenido
rpart.plot(model$finalModel)
```

Figura 24.3: Árbol de clasificación sin ajuste automático.

Este modelo se puede mejorar ajustando automaticamente los hiperparametros correspondientes a un modelo de árboles de decisión. De tal manera, primero es necesario saber los hiperparámetros a optimizar en el algoritmo implementado **R**.

```
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. El valor de `cp` es un parámetro de parada. Ayuda a acelerar la búsqueda de divisiones porque puede identificar las divisiones que no cumplen con este criterio y podarlas antes de ir avanzar por ella. También se puede entender `cp` como la mejora mínima necesaria en cada nodo del modelo. En busca del mejor valor de `cp` se define una red de posibles valores con los que evaluar el modelo.

```
# Se especifica un rango de valores típicos para los hiperparámetros
tuneGrid <- expand.grid(cp = seq(0, 0.1, 0.01))
```

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

```
model <- train(CLS_PRO_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
#>   0.00  0.9390141  0.8531746  0.8854497
#>   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
#>   0.06  0.7978269  0.9607143  0.6628307
#>   0.07  0.7978269  0.9607143  0.6628307
#>   0.08  0.7978269  0.9607143  0.6628307
#>   0.09  0.7866662  0.9321429  0.6664021
#>   0.10  0.7677249  0.8821429  0.6699735
#>
#> ROC was used to select the optimal model using the largest value.
#> The final value used for the model was cp = 0.
```

```
ggplot(melt(model$resample[, -4]), aes(x = variable, y = value, fill =
↪  variable)) +
  geom_violin()
```

De nuevo se puede ver el rendimiento de cada una de las métricas de los árboles entrenados utilizando validación cruzada. A continuación se muestra el árbol que genera este nuevo modelo.

Figura 24.4: Resultados modelo con ajuste automático durante la validación cruzada

```
# se pinta el árbol obtenido
rpart.plot(model$finalModel)
```

24.8.3. Árbol de regresión para estimar el número de días hospitalizado

En este ejemplo se utilizan los datos `cleveland`, los cuales estan contenidos en el paquete `CDR`, y que han sido utilizados en el capítulo **??** para estimar **¿dhosp?**. El conjunto de datos contiene información sobre pacientes que llegan a un hospital con dolor de pecho, de los que se han registrado distintas características. Se pretende predecir el número de días que un paciente necesitará de hospitalización en base al resto de caracteristicas observadas. Estas variables registran si el paciente esta diagnosticado de accidente coronario, su edad, su género, 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 <- train(
  dhosp ~ diag + edad + sexo +
    tdolor + dep,
  data = cleveland,
  method = "rpart",
  metric = "RMSE",
```

Figura 24.5: Árbol de clasificación con ajuste automático

```
  trControl = trainControl(method = "cv", number = 10)
)
```

```
model
#> CART
#>
#> 303 samples
#>   5 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 272, 273, 273, 272, 274, 272, ...
#> Resampling results across tuning parameters:
#>
#>   cp          RMSE      Rsquared   MAE
#>   0.01132433  1.508393  0.3168135  1.135381
#>   0.01674747  1.472518  0.3435459  1.103713
#>   0.37275022  1.576962  0.3526015  1.210831
#>
#> RMSE was used to select the optimal model using the smallest value.
#> The final value used for the model was cp = 0.01674747.
```

Los resultados durante la validación cruzada se observan en la figura **??**. Se puede comprobar que los resultados son bastante pobres, obteniendo un valor de R^2 que oscila entre un 20 % y un 55 %. Estos valores bajos de R^2 se corresponden con los altos valores obtenidos en RMSE.

```
ggplot(melt(model$resample[, -4]), aes(x = variable, y = value, fill =
↪  variable)) +
  geom_boxplot()
```

El resultado del modelo se muestra en el árbol de la figura **??**. La interpretación de este árbol sería:

1. Si el paciente no tiene diagnóstico de accidente coronario, solo necesita un día de hospitalización.
2. En el caso de tener este diagnostico, y una depresión mayor igual a dos en el segmento ST inducida por ejercicio en relación al reposo, necesitará 1,9 días de hospitalización.
3. En un último ejemplo, si la depresión en el segmento ST inducida por ejercicio en relación al reposo es mayor a 0,35 y menor 2 entonces se distingue por genero para saber el número de días hospitalizados. En el caso de los hombres, permaneceran 3,2 días hospitalizados. Mientras que las mujeres solo 1,9 días.

Figura 24.6: Resultados árbol de regresión durante la validación cruzada

```
# Se entrena de nuevo para obtener un objeto rpart
set.seed(101)
model <- rpart(dhosp ~ ., data = cleveland, cp = unlist(model$bestTune,
↪  use.names = FALSE))
# se pinta el árbol obtenido
rpart.plot(model)
```

Figura 24.7: Árbol de regresión para predecir el número de días de hospitalización

Resumen

En este capítulo se ha introducido al lector en los árboles de decisión para clasificación y regresión, en particular:

- Se ha presentado la lógica para la construcción de árboles de decisión, ya sean de regresión o clasificación.
- Se han contemplado 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 ha mostrado el uso de `R` para la clasificación de clases binarias, y para la predicción de variables respuesta numéricas a través de casos aplicados.

Capítulo 25

Máquinas de vector soporte

Ramón A. Carrasco e Itzcóatl Bueno[1]

25.1. Introducción

Aunque las maquinas 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 maquinas de vector soporte, SVM por sus siglas ingles *Support Vector Machines*, son similares a la regresión logistica pero la SVM enfatiza en el margen entorno a la frontera de decisión.

En la Figura 25.1 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 erroneas como podría pasar en el caso de la regresión logística y como se ilustra en la siguiente figura.

En resumen, los nuevos datos pueden ser clasificados dentro del margen, y cuanto mayor sea este, más margen de maniobra se tendrá para clasificar estos puntos. Por tanto, para obtener una clasificación erronea en la SVM es necesario que una observación se clasifique aún más al marge que en cualquier otro discriminante lineal. Otra ventaja de las SVM viene de la dificultad de clasificar perfectamente problemas del mundo real, pues es dificil que los discriminantes lineales, vistos en el capítulo **??**, logren una linea que divida perfectamente las categorias a clasificar. Sin embargo, en la SVM, se incluye en la funcion 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 ser divididos linealmente, no se cometerá

[1]Universidad Complutense de Madrid

Figura 25.1: SVM vs Regresión logística.

Figura 25.2: Nueva observación clasificada en SVM vs Regresión logística

ninguna penalización y se maximizará el margen. Mientras que si los datos no son linealmente separables, el mejor ajuste vendrá dado por el equilibrio entre una penalización del error total baja y un margen grande. La penalización a un punto mal clasificado es proporcional a la distancia desde el limite 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 alta. 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 se verá disminuido.

25.2. 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 parametros: 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:

$$\omega x - b = 0 \tag{25.1}$$

donde ωx es $\omega^{(1)}x^{(1)} + \omega^{(1)}x^{(1)} + \cdots + \omega^{(p)}x^{(p)}$ siendo p el número de variables incluidas en x. De este modo, la predicción para una instancia de x viene dada por:

$$y = sign(\omega x - b) \tag{25.2}$$

donde sign es el operador que devuelve +1 para cualquier valor positivo y -1 para los valores negativos. Entonces, el objetivo del algoritmo es obtener los valores óptimos ω^* y b^*.

Estos parametros se obtienen resolviendo un problema de optimización sujeto a las siguientes restricciones:

$$\begin{aligned} \omega x_i - b \geq 1 \text{ si } y_i = +1 \text{ y} \\ \omega x_i - b \leq 1 \text{ si } y_i = -1 \end{aligned} \tag{25.3}$$

Por otro lado, se busca maximizar el margen en torno a la frontera de decisión. Para conseguir esto es necesario minimizar la norma euclidea, y por tanto, el problema a resolver es:

$$\text{mín } ||\omega|| \text{ sujeto a}$$

$$y_i(\omega x_i - b) \geq 1 \text{ para } i = 1, \ldots, N$$

25.3. ¿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 tengo 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 gana la mayor de competiciones por pares.

25.4. Truco del Kernel: Tratando con la no linearidad

Cuando los datos no son linearmente separables, se extiende el modelo SVM incluyendo la función de perdida "hinge"(Gentile and Warmuth (1998); Lee and Lin (2013)), tal que $\text{máx}(0, 1-y_i(\omega x_i - b))$. En machine learning, esta función de perdida se utiliza para entrenar clasificadores, más concretamente para la clasificación por el margen máximo, sobre todo para las SVM. La función de perdida 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 clasificados, el valor obtenido con la función de perdida es proporcional a la distancia hasta la frontera de decisión. Entonces, se quiere minimizar la siguiente función de coste:

$$C||\omega||^2 + \frac{1}{N}\sum_{i=1}^{N} \text{máx}(0, 1 - y_i(\omega x_i - b)) \tag{25.4}$$

donde C es un hiperparametro que determina 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 opitimiza la función de perdida es llamado una SVM *soft-margin* mientras que el modelo original es denominado SVM *hard-margin*. Si se observa **?**(svm-loss) se puede ver que para valores grandes de C el segundo término es despreciable por lo que el algoritmo ignorará por completo la clasificación erronea 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 cometeran 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 transforme el espacio original a un espacio de mayor dimensión durante la optimización de la función de coste. Así, es posible transformar un espacio de datos bidimensional no separable linealmente en un espacio de datos tridimensional linealmente separable usando un mapeo especifico definido por $\phi : x \to \phi(x)$ donde $\phi(x)$ es un vector de mayor dimensionalidad 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.

Se puede trabajar eficientemente en espacios de mayor dimensión sin necesidad de hacer las trasformaciones explicitamente. El problema de optimización de una SVM puede ser reformulado como:

$$\max_{\alpha_1,\ldots,\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} y_i\alpha_i(x_i x_k)y_k\alpha_k \text{ sujeto a} \tag{25.5}$$
$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ y } \alpha_i \geq 0,\ i = 1,\ldots,N$$

donde α_i representa los multiplicadores de Lagrange. En esta formulación solo se aplica los datos de las variables en el término $x_i x_k$ asi que si se quiere transformar nuestros datos a una espacio de mayor dimension se necesita transformar x_i en $\phi(x_i)$ y x_k en $\phi(x_k)$ y después multiplicarlas. Sin embargo, esto es muy costoso.

Utilizando el truco del kernel se puede evitar este proceso costoso de transformación y evitarnos calcular el producto escalar reemplazandolo por una operación más simple con las variables originales que proporcione el mismo resultado.

25.4.1. Algunos kernels populares

Los kernels (Schölkopf et al. (1997); Scholkopf et al. (1997)) más populares en el entrenamiento de SVM estan incluidos dentro de la función `svm` del paquete `e1071` donde se puede determinar a través del parametro `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 hiperparametros, como δ o γ, que es necesario tunear. En machine learning se utiliza mucho el término **tunear**, el cual se refiere al hecho de ajustar autómaticamente tratando de optimizar los parametros del algoritmo. A la hora de ajustar en **R** un modelo SVM se puede conocer los parametros a ajustar utilizando la funcion `modelLookup`. Por ejemplo, para una SVM con kernel de base radial se observa:

```
caret::modelLookup("svmRadial")
#>       model parameter label forReg forClass probModel
#> 1 svmRadial     sigma Sigma   TRUE     TRUE      TRUE
#> 2 svmRadial         C  Cost   TRUE     TRUE      TRUE
```

Los dos parametros que se pueden ajustar en un modelo SVM con kernel de base radial en **R** son el coste (C) que es la constante C en la ecuación 25.4 y *sigma* que, en el caso de un modelo de regresión probabilística, es el parámetro de escala de la distribución de Laplace hipotética, con media cero, estimada por máxima verosimilitud.

25.5. Procedimiento con R: la funcion `svm`

En el paquete `e1071` de R se encuentra la función `svm` que se utiliza para entrenar un modelo maquinas vector soporte:

```
svm(x, y, scale = TRUE, type = NULL, kernel = ..., ...)
```

- 'x': Conjunto de datos de entrenamiento que contiene los predictores
- 'y': Un vector respuesta con las clases o valores de la variable respuesta.
- 'scale': Un valor booleano indicando si es necesario escalar las variables.
- 'type': Indicando si se pretende resolver un problema de clasificación o de regresión.
- 'kernel': El utilizado durante el entrenamiento y la predicción.

25.6. Aplicación de un modelo SVM Radial con ajuste automático en R

Los datos utilizados para entrenar el modelo SVM en este capitulo se cargan desde la libreria `CDR`. Además, para su entrenamiento se requieren las librerías `caret` y `e1071` que ya vienen incluidas en el paquete de este libro `CDR`.

```
library(CDR)

data(dp_entr_NUM)
```

Se contruye un modelo SVM con Kernel Radial usando el conjunto de entrenamiento con todas las variables transformadas a numéricas y aplicándole previamente a este conjunto de entrenamiento una normalización z-score, presentadas en el capítulo **??**, para que las distintas escalas numéricas sean equiparables. Además se va a proceder a ajustar de forma automática los parámetros más transcendentes de dicho algoritmo.

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

```
# Se especifica un rango de valores típicos 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 capítulo **??**, de 10 folds. Además, se le indica al modelo que debe calcular las probabilidad 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 sensitividad, especificidad y el area bajo la curva ROC. Finalmente, ante la necesidad de estandarizar los datos se le indica al modelo a través del argumento `preProcOptions`. Una vez tenemos definida una red de parametros a optimizar así como un procedimiento de muestreo para el entrenamiento se puede ajustar el modelo automáticamente con los parametros óptimos encontrados.

```
# Se fija una semilla común a todos los modelos
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_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 que 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 le puede indicar una métrica para el rendimiento del modelo, en caso de no indicarlo **R** asigna la más acorde al tipo de modelo. Finalmente, se incluyen las opciones de entrenamiento y la red de parámetros entre los que buscar la combinación óptima. Los parametros elegidos para el modelo entrenado son $\sigma = 0{,}1$ y $C = 0{,}1$ como puede observarse a continuación. Además de ver este resultado en la salida del modelo, se puede observar en la figura **??**. En este gráfico el eje y mide el rendimiento del modelo para ciertos valores de los parametros. Cada linea representa un valor para el hiperparámetro sigma, y se mide su rendimiento variando distintos nivels del parámetro coste (C), que queda representado en el eje x. Así, se observa que la linea roja (sigma=0,1) alcanza el mayor nivel de precisión en el valor C=0,1.

```
ggplot(model)
```

Los siguientes boxplot muestran el rendimiento del modelo en las distintas repeticiones del proceso de validación cruzada. De esta manera, vemos como la sensibilidad oscila en valores superiores al 75 % y la especificidad en valores superiores al 70 % lo cual indica que el modelo entrenado es capaz de predecir bastante bien tanto a los clientes que van a comprar el nuevo producto como los que no lo van a hacer.

Figura 25.3: Optimización de los parámetros C y sigma de una SVM.

```
ggplot(melt(model$resample[, -4]), aes(x = variable, y = value, fill =
↪  variable)) +
  geom_boxplot()
```

Figura 25.4: Resultados del modelo obtenidos durante la validación cruzada.

25.6.1. Importancia de las variables

Muchos algoritmos de machine learning de caja negra, no proporcionan información sobre que importancia tiene cada variable en el modelo. Este es el caso de la SVM. Sin embargo, utilizando

el paquete `vip` podemos cuantificar la importancia de cada variable. Otros paquetes de R como son `DALEX` o `iml` pueden hacer esto mismo.

Para medir la importancia, la función debe reflejar que la metrica para entrenar el modelo fue el area bajo la curva (AUC). La función de medida que se aporte a `vip()` en el parametro `pred_wrapper` deber contener los argumentos de valores reales observados y valores predichos. La SVM entrenada utiliza AUC para el ajuste del modelo, por lo que función aportada a `vip` deberá devolver la probabilidad de que el modelo asigne una observación a la clase de referencia. En el ejemplo mostrado, la clase de referencia es "SI", puesto que interesa saber si un cliente comprará el nuevo producto. Entonces, la función de predicción viene dada por:

```
prob_si <- function(object, newdata) {
  predict(object, newdata = newdata, type = "prob")[, "S"]
}
```

Finalmente, si se ejecuta la función `vip()` aportandole los argumentos necesarios generará la figura **??**. Este gráfico indica la importancia de cada variable en el modelo de más a menos importante. En el ejemplo mostrado, 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 de las variables mostradas tener un nivel de educación básico no es muy relevante 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")
)
```

Así, de la figura **??** se concluye 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 *smartchwatch fitness*, si compró o no el *smartchwatch 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.

Figura 25.5: Importancia de las variables incluidas en la SVM

Resumen

En este capítulo se ha introducido al lector en el algoritmo de maquinas vector soporte, en particular:

- Se presenta el concepto de margen de decisión, y las ventajas de la SVM respecto a otras técnicas de clasificación por incorporarlo.
- Se explica el truco del kernel cuando los datos no son separables por un hiperplano en su espacio original. Además, son mencionados algunos de los kernels más utilizados.
- Se presentan casos aplicados de una SVM con kernel radial en `R` para la clasificación de datos con respuesta binaria, y se presenta el caso extendido en el que se ajustan automáticamente los hiperparametros.
- Se contempla la importancia de las variables en algoritmos de caja negra.

Capítulo 26

Clasificador k-vecinos más próximos

Ramón A. Carrasco e Itzcóatl Bueno[1]

26.1. Introducción

El k-vecinos más próximos (KNN, por sus siglas inglesas de k-nearest neighbors) es un algoritmo de aprendizaje no paramétrico. Al contrario que otros algoritmos de aprendizaje que permiten deshacerse de los datos de entrenamiento una vez el modelo es construido, KNN guarda las observaciones de entrenamiento en memoria. Cuando se cuenta con una nueva observación x el algoritmo KNN encuentra las k observaciones de entrenamiento más cercanas a la nueva y devuelve la clase mayoritaria (en el caso de clasificación) o el valor medio (en el caso de regresión).

Determinar el número de puntos, k, que se van a utilizar para clasificar los nuevos puntos es crucial (James et al. (2013)). Si por ejemplo, se determina que $k = 3$, el modelo KNN utilizará los tres puntos más cercanos (vecinos) a la nueva observación para clasificarlo. El número de vecinos elegidos es decisivo para la determinación del resultado. Por ello, se recomienda probar multiples combinaciones de k para dar con el mejor ajuste y evitar valores muy bajos o muy altos de k. Si se establece un valor muy bajo de k aumentara el sesgo y llevará a clasificaciones erroreas. Mientras que valores muy elevados de k harán que el algoritmo sea computacionalmente costoso. Además, es recomendable establecer valores impares de k para evitar puntos muertos estadisticos y un resultado no valido.

Dado que la escala de las variables puede impactar en el resultado del modelo KNN, el conjunto de datos debe escalarse para estandarizar la varianza. Aunque el modelo KNN es facil de entender y generalmente preciso, almacenar todo un conjunto de datos y clacular la distancia entre

[1]Universidad Complutense de Madrid

Figura 26.1: Ejemplo de k-vecinos más cercanos

cada nuevo punto a clasificar y todos los puntos del conjunto de datos supone una necesidad de recursos informáticos muy elevados. 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 muy lentos. Por este motivo, no se recomienda el uso del algoritmo knn cuando se dispone de conjuntos de datos muy grandes. Otro inconveniente 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 puntos con multiples dimesiones también incrementaría la necesidad de recursos informáticos y podría dificultar conseguir una clasificación precisa.

26.2. Decisiones a tener en cuenta

La elección de la función de distancia así como el número de vecinos k son elegidos por el investigador, como ya se ha mencionado, se hace antes de ejecutar el algoritmo. Siendo este último el hiperparametro del modelo.

```
caret::modelLookup("knn")
#>   model parameter      label forReg forClass probModel
#> 1   knn         k #Neighbors   TRUE     TRUE      TRUE
```

26.2.1. Función de distancia a utilizar

La forma en que el modelo KNN determina la cercanía entre dos observaciones es con una función de distancia. Generalmente se utilizan la distancia euclidea, mostrada en la ecuación 26.1, y la distancia de Manhattan, cuya formula es la mostrada en la ecuación 26.2. Otras distancias

como las presentadas en el capítulo **??** también pueden ser utilizadas para el entrenamiento de este algoritmo.

$$d(x_i, x_k) = \sqrt{\sum_{j=1}^{p} (x_i^{(j)} - x_k^{(j)})^2} \tag{26.1}$$

$$d(x_i, x_k) = \sum_{j=1}^{p} |x_i^{(j)} - x_k^{(j)}| \tag{26.2}$$

Otra función de distancia utilizada es la similitud de coseno negativa, definida por:

$$s(x_i, x_k) = \cos(\angle(x_i, x_k)) = \frac{\sum_{j=1}^{p} x_i^{(j)} x_k^{(j)}}{\sqrt{\sum_{j=1}^{p} (x_i^{(j)})^2}\sqrt{\sum_{j=1}^{p} (x_k^{(j)})^2}} \tag{26.3}$$

Esta función mide la similitud en las direcciones de dos vectores. Si el ángulo entre dos vectores es cero, entonces esos dos vectores tienen la misma dirección y la similitud del coseno es igual a 1. En caso de que dos vectores vayan en direcciones opuestas, la similitud del coseno será -1. Si se quiere utilizar esta similitud como métrica de distancia es necesario multiplicarla por -1.

Finalmente, en el caso de querer incluir variables de distintos tipos en el cálculo de la distancia la opción más a adecuada es el coeficiente de disimilitud de Gower. Esta distancia se define como:

$$d(i, j) = \frac{\sum_{k=1}^{p} \omega_k \delta_{ij}^{(k)} d^(k)_{ij}}{\omega_k \delta_{ij}^{(k)}} \tag{26.4}$$

En resumen, es una media ponderada de las distancias $d_{ij}^{(k)}$ con ponderaciones $\omega_k \delta_{ij}^{(k)}$.

26.2.2. Número de vecinos (k) seleccionados

La elección de cuantos vecinos (k) intervienen en el ajuste del algoritmo es determinante para su rendimiento. Si se escogen demasiados pocos vecinos el modelo sobreajusta. El caso más extremo, en el que elegimos $k = 1$ se basa la predicción en la observación con la distancia más cercana. Por otro lado, eligiendo un número alto de vecinos hace que el modelo no ajuste bien. En este sentido, en el caso extremo de elegir todas las observaciones como vecinos más cercanos, es decir, $k = n$, 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 optima de k, puesto que en gran medida depende del conjunto de datos utilizado. Cuando se trabaja con conjuntos de datos que tienen pocas variables que no aporten información, valores pequeños de k tienden a funcionar mejor. Cuantas más variables sin importancia se incorporan en el conjunto de datos, mayores deberan ser los valores de k para suavizar su efecto.

26.3. Procedimiento con R: la funcion knn

En el paquete class de R se encuentra la función knn que se utiliza para obtener 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': Factor con las autenticas clases de las observaciones de entrenamiento.
- 'k': Número de vecinos a considerar

26.4. Aplicación del modelo KNN en R

Se va a implementar un modelo KNN para la clasificación de clientes que compren o no el *tensiómetro digital* teniendo en cuenta sus caracteristicas y el resto de sus compras. Este conjunto de datos viene incluido el paquete `CDR`, y es la versión numérica del conjunto de datos `dp_entr`. Estos datos tienen sus variables en escala numérica, posibilitando el cálculo de las distancias para el modelo KNN. Sin embargo, las variables tienen distintas escalas de medida (euros, años, unidades, etc.) por lo que es necesario indicar en la función `trainControl` que se haga un preprocesamiento para estandarizar las variables. Además, se define que el método de muestreo sea una validación cruzada, como la presentada en el capítulo **??**, con 10 folds.

```
library(CDR)
data(dp_entr_NUM)

# Definimos un método de remuestreo
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5,
  classProbs = TRUE,
  preProcOptions = list("center"),
  summaryFunction = twoClassSummary
)
```

Antes de entrenar el modelo, es necesario la selección del número óptimo de vecinos k. Evaluar uno a uno los posibles valores sería un trabajo arduo, por ello en **R** se puede definir una red de posibles valores sobre los que evaluar el modelo KNN, y que seleccione el que mejor rendimiento proporcione. A continuación se determinan los posibles valores de k que se quieren evaluar, en este caso todas las opciones entre uno y quince vecinos.

```
# Definimos la red de busqueda del parámetro
hyper_grid <- expand.grid(k = 1:15)
```

Una vez se definen tanto el método de muestreo como la red de posibles valores del hiperparámetro se puede entrenar el modelo. Se procede con el ajuste automático del modelo de k vecinos más próximos. A la vez que se determina el número óptimo de vecinos, se entrena el modelo final para que puede ser utilizado con futuras observaciones.

```
set.seed(101)
# Entrenamos el modelo buscando el parámetro 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
#>  19 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  0.6280952  0.6309259
#>    3  0.6828893  0.6216931  0.6489153
#>    4  0.6851087  0.6223545  0.6401852
#>    5  0.6951129  0.6540212  0.6651852
#>    6  0.6914664  0.6173810  0.6616138
#>    7  0.6982592  0.6252381  0.6953439
#>    8  0.6974556  0.6223545  0.6982275
#>    9  0.6992229  0.6145238  0.7110582
#>   10  0.6994133  0.6123545  0.7124868
#>   11  0.6980371  0.5843651  0.7232011
```

```
#>   12  0.6957729  0.5851323  0.7188889
#>   13  0.6965353  0.5794180  0.7246825
#>   14  0.6910927  0.5664815  0.7189418
#>   15  0.6875879  0.5763492  0.7232011
#>
#> ROC was used to select the optimal model using the largest value.
#> The final value used for the model was k = 10.
```

Además, este mismo resultado se puede mostrar graficamente para facilitar su presentación a terceros o facilitar la comprensión de por qué se lleva a cabo esa decisión. Por ejemplo, en la figura **??** se observa que el número óptimo de vecinos es $k = 10$, donde se alcanza el mejor rendimiento del modelo.

```
ggplot(model) +
  geom_vline(xintercept = unlist(model$bestTune), col = "red", linetype =
  ↪  "dashed") +
  theme_light()
```

Figura 26.2: Número óptimo de vecinos (k)

El boxplot de los resultados obtenidos durante el proceso de validación cruzada muestra que el ROC del modelo oscila entre un 55 % y un 85 % aproximadamente.

El modelo se resiente en su rendimiento pues parece mostrar dificultades para ajustar la clase positiva, al tener valores de sensibilidad de entre el 40 % y el 85 %, como puede observarse en la figura **??**; resultados ligeramente peores que los que obtiene al ajustar observaciones de la clase negativa, los cuales oscilan entren el 54 % y el 89 %.

Figura 26.3: Resultados obtenidos durante el proceso de validación cruzada

Resumen

En este capítulo se ha introducido al lector en el algoritmo de aprendizaje supervisado conocido como k-vecinos más próximos, destacando:

- Las decisiones a tener en cuenta antes de proceder a entrenar un modelo de k-vecinos más próximos.
- Se han planteado algunas de las distancias más utilizadas para el entrenamiento de este tipo de modelos.
- Se han definido los pros y contras de la cantidad de vecinos a tener en cuenta, así como su ajuste automático.

Capítulo 27

Naive Bayes

Ramón A. Carrasco e Itzcóatl Bueno[1]

27.1. Introducción

Mientras que los algoritmos hasta ahora eran un conjunto de reglas para llevar a cabo un cálculo, un algoritmo bayesiano es un conjunto de reglas para usar la evidencia de tus datos para cambiar tus creencias. Los metodos bayesianos pueden formularse como un algoritmo (Lancaster (2004)) con los siguientes pasos:

1. Formular un modelo como una colección de distribuciones de probabilidad condicional a diferentes valores para un parámetro del modelo $\theta \in \Theta$.
2. Organizar las creencias sobre θ en una distribución de probabilidad a priori de θ.
3. Introducir los datos en la familia de distribuciones proporcionada en el paso 1.
4. Usa el teorema de Bayes para calcular tus nuevas creencias sobre θ.
5. Critica tu modelo.

Como puede observarse, en esta formulación de un algoritmo bayesiano es necesario el uso del **Teorema de Bayes** y en el caso del modelo Naive Bayes también es así.

27.2. 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:

[1]Universidad Complutense de Madrid

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{27.1}$$

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:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{27.2}$$

Esta formula es la definición del teorema de Bayes. El algoritmo de clasificación Naive Bayes (NB) esta basado en este teorema. Los conceptos de estadística aquí presentados pueden consultarse en más detalle en el capítulo **??**.

27.3. El algoritmo Naive Bayes

Si se adapta el teorema de Bayes a un problema de clasificación, tendríamos:

$$P(C = c|\ell) = \frac{P(\ell|C = c) \cdot P(C = c)}{P(\ell)} \tag{27.3}$$

En este caso, el elemento a la derecha del igual representa el objetivo de estimación en un problema de clasificación, es decir, la probabilidad de que un individuo pertenezca a la clase c despues de haber observado la evidencia ℓ (las variables incluidas en el modelo). Esta es la denominada probabilidad a posteriori. El resto de elementos de la formula, los que se ven a la izquierda del igual se definen a continuación:

- $P(C = c)$ 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 que clase pertnezca el individuo.

Sin embargo, una gran dificultad para aplicar esta ecuación es la necesidad de conocer que $P(\ell|c)$ es igual a $P(\ell_1 \cap \ell_2 \cap \cdots \cap \ell_\kappa|c)$. Esto es dificil de medir y muy especifico. Es complicado que encontremos un ejemplo concreto en nuestros datos de entrenamiento que coincida a la perfección con ℓ, y en el caso de tenerlo no tendríamos suficientes para poder estimar una probabilidad de forma fiable. La forma de solucionar este problema es incluir una suposición de independencia particularmente fuerte.

El concepto de indenpendencia condicional implica que conocer un evento no aporta información sobre otro evento. Esto puede expresarse de la siguiente forma:

$$P(AB|C) = P(A|C) \cdot P(B|C) \tag{27.4}$$

De este modo, volviendo a nuestro problema de clasificación en el que teniamos la dificultad de estimar $P(\ell_1 \cap \ell_2 \cap \cdots \cap \ell_\kappa|c)$, ahora tendriamos que:

$$P(\ell|c) = P(\ell_1|c) \cdot P(\ell_2|c) \cdots P(\ell_\kappa|c) \tag{27.5}$$

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 obtenemos la ecuación dada por el modelo Naive Bayes:

$$P(c|\ell) = \frac{P(\ell_1|c) \cdot P(\ell_2|c) \cdots P(\ell_\kappa|c)P(c)}{P(\ell)} \tag{27.6}$$

El clasificador Naive Bayes clasifica una nueva observación estimando la probabilidad de que pertenezca a cada clase y asignandolo a aquella que tenga la mayor probabilidad.

En definitiva, el clasificador Naive Bayes es muy eficiente en terminos de espacio de almacenamiento necesario así como tiempos de procesamiento. Además, a pesar de ser muy simple tiene en cuenta las caracteristicas 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 disponde de nuevas observaciones.

El ejemplo presentado en el capítulo 24 en el que se buscaba predecir si se podría jugar o no al tenís bajo unas condiciones meteorológicas concretas, puede desarrollarse utilizando el modelo Naive Bayes. En este caso, el procedimiento puede resumirse en tres pasos:

1. Resumir los datos en una tabla de frecuencias.
2. Generar una tabal de verosimilitud obteniendo las probabilidades de las variables.
3. Aplicar el teorema de Bayes para calcular la probabilidad a posteriori.

De este modo, en caso de tener 30 observaciones que registrasen la previsión del día (soleado, nublado, lluvioso) y si ese día se jugó o no, y no se contase con información sobre niebla ni viento, el primer paso sería resumir estos datos en una tabla de frecuencias como la tabla 27.1.

En un segundo paso, se obtienen las probabilidades de cada posibilidad a partir de la Tabla 27.1 y resultando en la tabla 27.2.

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 prevee soleado? De acuerdo al teorema de Bayes, esta pregunta se respondería a través de:

$$P(No|Soleado) = \frac{P(Soleado|No) \cdot P(No)}{P(Soleado)}$$

	¿Se jugó?		
Clima	SÍ	NO	Total
Soleado	10	1	11
Nublado	5	5	10
Lluvioso	2	7	9
Total	17	13	30

Tabla 27.1: Tabla de frecuencias - Clima vs Partidos jugados

	¿Se jugó?		
Clima	SÍ	NO	$P(Clima_i)$
Soleado	10	1	$\frac{11}{30} = 0,37$
Nublado	5	5	$\frac{10}{30} = 0,33$
Lluvioso	2	7	$\frac{9}{30} = 0,30$
$P(Jugar)$	$\frac{17}{30} = 0,57$	$\frac{13}{30} = 0,43$	

Tabla 27.2: Tabla de verosimilitud - Clima vs Partidos jugados

A partir de la tabla 27.2 se pueden obtener todos los elementos de este ejemplo. Así, se tendría:

$$
\begin{aligned}
P(Soleado|No) &= \frac{1}{13} = 0,08 \\
P(Soleado) &= 0,37 \\
P(No) &= 0,43
\end{aligned}
$$

Entonces, la probabilidad de no jugar, es decir, de asignar el día como no jugable dado que la previsión del clima es soleado es igual a 0,09 de acuerdo a la ecuación 27.7.

$$
P(No|Soleado) = \frac{0,08 \cdot 0,43}{0,37} = 0,09 \tag{27.7}
$$

27.4. Procedimiento con R: la funcion `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': Que refleja el predictor lineal $Y \sim X_1 + ... + X_p$.
- 'data': Un conjunto de datos con el que interpretar el nombre de las variables incluidas en la formula.

- 'prior': Vector con las probabilidades a priori de las clases.

27.5. Aplicación del modelo Naive Bayes

Se contruye el modelo Naive Bayes usando el conjunto de entrenamiento de compras realizadas por clientes incluido en el paquete `CDR`. Este conjunto de datos tiene unos predictores que indican que productos han comprando los clientes, el importe que han gastado y otras características como su edad o 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(CDR)
data("dp_entr")
```

```
# se fija una semilla común a todos los modelos
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_pro13 ~ .,
  data = dp_entr,
  method = "nb",
  metric = "Accuracy",
  trControl = trainControl(
    classProbs = TRUE,
    method = "cv",
    number = 10
  )
)
```

```
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
#>
#> 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 para el argumento `usekernel` es indiferente indicarlo en FALSE o TRUE, los resultados de precisión son similares. En caso de indicar que se quiere usar, se utiliza la densidad para estimar las densidades condicionales de clase de los predictores métricos. El resto de hiperparámetros no se han ajustado al determinar un valor constante para este ejemplo. Finalmente, la salida muestra que la precisión media obtenida durante la validación cruzada alcanza un 85,1 % indicando 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 observaque en media el modelo predice mejor cuando un cliente no va a comprar que cuando si lo hace, aunque no con mucha diferencia (menos de un 2 %).

```
# resultados del modelo por cada iteración del cv
ggplot(melt(model$resample[, -4]), aes(x = variable, y = value, fill =
↪  variable)) +
  geom_boxplot()
```

Los resultados obtenido durante la validación cruzada se han guardado y se puede observar como la precisión oscila entre el 76 % y el 91 %, aunque en uno de los resultados se obtuvo un 96 % de precisión el cual se marca como un resultado atípico.

Figura 27.1: Resultados del modelo Naive Bayes obtenidos durante el proceso de validación cruzada.

Resumen

En este capítulo se ha introducido al lector en el algoritmo de Naive Bayes, en concreto:

- Se han presentado los fundamentos del algoritmo bayesiano, particularmente el Teorema de Bayes.
- Se ha explicado el funcionamiento del algoritmo Naive Bayes y su relación con el teorema de Bayes.
- Se ha demostrado su aplicabilidad a casos reales de clasificación a traves de `R`.

Capítulo 28

Bagging. Random Forest

Ramón A. Carrasco e Itzcóatl Bueno[1]

28.1. Introducción: Metodos de Ensamble

A veces puede darse el caso en que ninguno de los modelos hasta ahora presentados proporciona resultados convincentes para nuestro problema. El *aprendizaje ensamblado* Zhou (2012) es un paradigma que en lugar de entrenar un modelo muy preciso se centra en entrenar un gran número de modelos con baja precisión y después combinar sus predicciones para obtener un metamodelo una precisión más alta.

Los modelos de baja precisión son entrenados por algoritmos débiles, es decir, algoritmos incapaces de aprender modelos complejos; y por tanto, generalmente son rápidos tanto en tiempos de entrenamiento como de procesamiento. Existen dos paradigmas de aprendizaje ensamblado: el *bagging* y el boosting.

28.2. Bagging

En vez de buscar la división más eficiente en cada capa como ocurre en el árbol de decisión, una alternativa sería construir multiples árboles de decisión y combinar sus resultados. Esta técnica se conoce como bagging y consiste en hacer crecer varios árboles utilizando una selección aleatoria de los datos que se usan para cada árbol y combinando 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.

La principal característica del bagging es el llamado muestreo bootstrap. La intuición tras esto es que para que los árboles generen una respuesta única, debe existir aleatoriedad y variación en

[1]Universidad Complutense de Madrid

cada árbol que conforme el modelo final; puesto que no tendría sentido construir varios árboles identicos. Este problema queda resuelto por el muestreo bootstrap, el cual extrae una variación aleatoria de los datos en cada ronda. En el caso del bagging, se ejecutan distintas muestras de datos para el entrenamiento de cada árbol. A pesar de que esto no elimina la problematica del sobreajuste, los patrones presentes en el conjunto de datos apareceran en la mayoría de los árboles entrenados y aparecerán en la predicción final. Por tanto, el bagging es una técnica de gran eficacia para el tratamiento de los valores atipicos y para la reducción de la varianza que generalmente afecta a un único árbol de decisión.

28.3. 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': Que refleja el predictor lineal $Y \sim X_1 + ... + X_p$.
- 'data': Un conjunto de datos con el que interpretar el nombre de las variables incluidas en la formula.

28.4. Implementando bagging en R

Es posible la implementación de un modelo de predicción de agregación bootstrap en R. Para ello, se pueden utilizar multiples funciones como la ya mencionada en la sección 28.3 `bagging`.

```
library(CDR)
data("dp_entr")
```

```
# se fija una semilla para que el modelo sea reproducible
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
```

Desafortunadamente, `bagging()` no proporciona el error de clasificación incorrecta por árbol, por lo que para obtener una curva de error mostrada en la figura 28.1 se itera el modelo variando los valores del parametro `nbagg` entre 10 y 150, incrementandolo de cinco en cinco. Se observa que el error mínimo se obtiene al utilizar 60 árboles.

Figura 28.1: Número de árboles óptimo

El modelo se puede optimizar ajustando a sus hiperparametros óptimos. Esto se debe a que el bagging tambien está incluido en la función `caret` y por tanto podemos saber qué parametros son los que hay que ajustar. Además, se puede entrenar utilizando validación cruzada para comprobar si el modelo se puede generalizar. Se observa que si se entrena un modelo bagging con 60 árboles, la precisión del modelo es del 87 %. Esto es equivalente al resultado obtenido anteriormente en el que para 60 árboles el modelo tenía un error de clasificación del 13 %.

```
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
```

28.4.1. Interpretación de variables en el bagging

Una de las principales desventajas del bagging es que los modelos que son intepretables, tras el uso del bagging ya no lo son. A pesar de esto, todavía se puede hacer inferencia de como cada una de las variables influye en el modelo generado. La forma de medir la importancia de las variables incluidas en un árbol es tabular la reducción de la función de perdida atribuida en cada particióna cada variable. Una variable puede utilizarse varias veces para dividir el árbol y por tanto, se suma la reducción total en la función de perdida en todas las divisiones de esa variable, utilizandose como su importancia total. Este proceso es similar para el bagging. En este caso, para cada árbol se cácula la reducción de la función de pérdida en todas lasdivisiones. Tras esto, se agrega esta medida en todos los árboles para cada función. 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` si lo hace y se puede construir un gráfico de importancia usando la función `vip`.

```
library(vip)
vip(model_bag,
  num_features = 15,
```

```
  aesthetics = list(color = "skyblue", fill = "skyblue")
)
```

Figura 28.2: Importancia de las variables incluidas en el modelo bagging.

La figura **??** muestra que las variables más importantes en el modelo bagging entrenado para predecir si un cliente comprará o no el *tensiómetro digital* son si comprará la *depiladora eléctrica* y cuánto importe gastará en ese producto, seguido de si comprará el *estimulador muscular* y el *smartchwatch fitness*.

28.5. Random Forest

El bagging es el paradigma tras el algoritmo de random forest. Este algoritmo fue desarrollado por primera vez por Tin Kam Ho en 1995 Ho (1995). Sin embargo, fueron Cutler Cutler and Zhao (1999) y Breiman Breiman (2001) quienes desarrollaron una versión extendida del modelo y registraron **random forest** como marca comercial. El algoritmo básico de bagging funciona del siguiente modo: a partir del conjunto de entrenamiento, se generan K muestras aleatorias $\mathbb{S}_k$ y se entrena un modelo de árbol de decisión f_k utilizando cada muestra $\mathbb{S}_k$ como el conjunto de entrenamiento. Tras el entrenamiento, se dispone de K árboles de decisión. La predicción de una nueva observación x se obtiene como la media de las K predicciones:

$$y \leftarrow \hat{f}(x) = \frac{1}{K}\sum_{k=1}^{K} f_k(x) \tag{28.1}$$

en el caso de regresión, o por la mayoría de votación en el caso de clasificación.

Tanto el bagging como el random forest desarrollan multiples 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 partición.

Figura 28.3: Ejemplo de Random Forest

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 predicciones dadas por los árboles esten altamente correlacionadas. El modelo random forest evita este problema al establecer la obligación en cada división de utilizar un subconjunto de las variables, lo que 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. Finalmente, aunque estos modelos funcionan bien en la interpretación de patrones complejos y son versatiles, otras técnicas, como por ejemplo el gradient boosting, proporcionan una mayor precisión en las predicciones.

Estos modelos se han vuelto populares porque tienden a proporcionar un muy buen rendimiento con los modelos predeterminados. A pesar de tener muchos hiperparametros que pueden ser ajustados, los valores por defecto de estos tienden a ofrecer buenos resultados en la predicción. Los hiperparametros más importantes que hay que ajustar al entrenar un modelo random forest son: el número de árboles (K), el numero 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.

28.5.1. Número de árboles (K)

El primer parametro que la lógica nos lleva a ajustar es el número de árboles que componen el modelo de random forest. Aunque no se considera un hiperparametro como tal, es necerio ajustar el número de árboles a utilizar, pues 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 emabrgo, 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 obtendran estimaciones de error más estables.

28.5.2. Número de variables a considerar ($mtry$)

$mtry$ se refiere al hiperparametro encargado de controlar la aleatorización de variables utilizadas para las particiones de los árboles. Este hiperparametro ayuda a equilibrar la baja correlación del árbol con una razonable fuerza predictiva. Existe un valor predeterminado para este hiperparametro 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 los 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 $mtry$.

28.5.3. Complejidad de los árboles

La base de un random forest es que se construye con árboles de decisión, de los cuales se puede controlar su profundidad y complejidad como se vió en el capítulo 24. Esto se puede hacer ajustando los hiperparametros de profundidad máxima permitida, tamaño del nodo o la cantidad máxima de nodos terminales, por ejemplo.

El tamaño del nodo es probablemente el hiperparámetro más común para controlar la complejidad del árbol y la mayoría de las implementaciones usan los valores predeterminados de uno para la clasificación y cinco para la regresión, ya que estos valores tienden a producir buenos resultados. Si se quiere controlar el tiempo de procesamiento, se puede hacer reducciones significativas del tiempo aumentando el tamaño del nodo y esto solo impactará marginalmente en la estimación del error.

28.5.4. Esquema de muestreo

Por defecto, el random forest tiene como esquema de muestreo el bootstrapping, en el cual todas las observaciones se muestrean con reemplazo. Esto es que todas las copias 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 si se muestrea con o sin reposición. El parámetro 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 optimo puede hayarse evaluando tres o cuatro valores que oscilen entre el 25 % y el 100 % y en el caso de que haya variables no balanceadas categóricas se puede intentar muestrear sin reposición.

28.5.5. 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 capítulo 24. Esto es, en el caso de regresión seleccionar la división que minimiza la suma residual de cuadrados (SRC); y en el caso de clasificación la división que minimiza la impureza de Gini o la entropía.

28.6. Procedimiento con R: la función `randomForest`

En el paquete `randomForest` de **R** se encuentra la función `randomForest` que se utiliza para entrenar un modelo random forest:

```
randomForest(formula, data = ..., ...)
randomForest(x, y, xtest, ytest, ntree = 500, mtry, ...)
```

- 'formula': Que refleja el predictor lineal $Y \sim X_1 + ... + X_p$.

- 'data': Un conjunto de datos con el que interpretar el nombre de las variables incluidas en la formula.
- 'x': Conjunto de datos de entrenamiento que contiene los predictores
- 'y': Un vector respuesta con las clases o valores de la variable respuesta.
- 'xtest': Conjunto de datos que contiene los predictores del dataset de validación.
- 'ytest': Variable respuesta del conjunto de validación.
- 'ntree': Número de arboles a construir en el modelo.
- 'mtry': Numero de variables muestreadas aleatoriamente como candidatas en cada partición.

28.7. 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_NUM`. Estos datos tienen las variables categóricas de `dp_entr` ya convertidas en variables one-hot-enconding. Se busca predecir si un cliente va a comprar o no el nuevo producto de acuerdo a que productos compra, el importe que gasta en ellos y otras características como, por ejemplo, su nivel educativo.

```
library(CDR)
data(dp_entr_NUM)
```

28.7.1. Aplicación del Random Forest

Este algoritmo al estar basado en árboles de clasificación tiene los mismos requisitos para el entrenamiento que tenían dicho árboles, así se contruye el modelo usando el conjunto de entrenamiento.

```
# se fija una semilla común a todos los modelos
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_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 hiperparametro *mtry* se ha ajustado a 10 variables.

Figura 28.4: Resultados del modelo random forest durante el proceso de validación cruzada.

Finalmente, aunque el random forest generado esta compuesto por 500 árboles, se puede acceder a cualquiera de ellos para estudiarlos en profundidad. Para ello, es necesario instalar el paquete `reprtree` el cual se encuentra en un repositorio de github.

```
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 que variable se utiliza para la partición, cuál es el valor que decide la división, indicando si es un nodo terminal o no y la predicción del nodo, el cual es `NA` si no es un nodo terminal.

```
set.seed(101)
rf <- randomForest(CLS_PRO_pro13 ~ .,
  data = dp_entr_NUM, ntree = 500,
  mtry = unlist(model$bestTune)
)

# Observamos 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>
#> 5 importe_pro17         150      1       <NA>
#> 6      anos_exp          33      1       <NA>
```

Este árbol se puede mostrar de la siguiente manera. Sin embargo, el método por el que se representa gráficamente no es muy claro y puede llevar a confusión o a dificultar la interpretación del árbol. Si se desea estudiar hasta cierto nivel del árbol, se puede incluir el parametro `depth` como en el ejemplo abajo mostrado.

```
library(reprtree)
plot.getTree(rf, k = 205)
```

```
plot.getTree(rf, k = 205, depth = 5)
```

28.7.1.1. Aplicación del modelo Random Forest con ajuste automático

En este segundo ejemplo, se pretenden mejorar los resultados del modelo anterior. Para ello, se va a proceder a ajustar de forma automática los parámetros más transcendentes de dicho

Figura 28.5: Árbol número 205 del random forest entrenado.

Figura 28.6: Árbol número 205 del random forest entrenado hasta la capa 5.

algoritmo. De los mencionados anteriormente, solo vamos a intentar ajustar el *mtry*, que es el que incluye el método `rf` como hiperparametro.

```
modelLookup("rf")[, 1:3]
#>   model parameter                         label
#> 1    rf      mtry #Randomly Selected Predictors
```

Para ajustar el número de árboles, y el resto de hiperparametros se podría hacer iterando el modelo y cambiando sus valores. Los valores que se quieren probar para el parametro *mtry* se incluyen en una red de opciones que se determina a continuación.

```
# Se especifica un rango de valores típicos para los hiperparámetros
tuneGrid <- expand.grid(mtry = 2:19)
```

A continuación se entrena el modelo para que lo ajuste a los valores de los hiperparametros que maximicen el rendimiento predictivo del modelo.

```
# se fija una semilla común a todos los modelos
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_pro13 ~ .,
  data = dp_entr_NUM,
  method = "rf", metric = "Accuracy",
  tuneGrid = tuneGrid,
  trControl = trainControl(classProbs = TRUE)
)
```

```
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
#>    2    0.8641273  0.7280186
#>    3    0.8645144  0.7287607
#>    4    0.8618664  0.7234885
```

```
#>     5     0.8630134  0.7257459
#>     6     0.8600585  0.7198120
#>     7     0.8593013  0.7182988
#>     8     0.8629885  0.7256405
#>     9     0.8589407  0.7175065
#>    10     0.8567871  0.7132165
#>    11     0.8588108  0.7172628
#>    12     0.8595203  0.7187110
#>    13     0.8579929  0.7156347
#>    14     0.8581897  0.7160362
#>    15     0.8585831  0.7168220
#>    16     0.8581382  0.7159461
#>    17     0.8578179  0.7152923
#>    18     0.8579168  0.7155132
#>    19     0.8569240  0.7135054
#>
#> Accuracy was used to select the optimal model using the largest value.
#> The final value used for the model was mtry = 3.
```

Ahora que la busqueda del valor óptimo *mtry* ha sido más exhaustiva que en el ejemplo se observa que los resultados con 3 variables seleccionadas en cada partición es suficiente, y que no son necesarias 10 como en el ejemplo anterior pues la precisión decae. Finalmente, se puede observar en el boxplot los resultados obtenidos durante la validación cruzada. Se observa cómo no solo la precisión es mayor que en el ejemplo anterior, sino que además los resultados tienen menos dispersión.

Figura 28.7: Resultados obtenidos por el random forest con ajuste automático durante el proceso de validación cruzada.

Resumen

En este capítulo se ha introducido al lector en el algoritmo de aprendizaje supervisado conocido como random forest, en concreto:

- Se ha presentado el concepto de aprendizaje ensamblado, y profundizado en uno de sus paradigmas: el bagging.
- Se ha implementado el bagging en `R` a través de un caso de aplicación para la clasificación binaria de datos.
- Se ha expuesto como medir la importancia de las variables incluidas en un modelo bagging para facilitar su interpretación.
- Se ha explicado el modelo random forest, fundamentado en los árboles decisión y en el bagging. Así como los hiperparámatros más importantes para ajustar un modelo con el mejor rendimiento.
- Se ha aplicado un ejemplo de clasificación binaria utilizando el modelo random forest en `R`.

Capítulo 29

Boosting. XGBoost

Ramón A. Carrasco e Itzcóatl Bueno[1]

29.1. Introducción. Boosting.

El boosting Schapire and Freund (2012) es el otro de los paradigmas de aprendizaje ensamblado presentado en el capítulo 28. De nuevo, el boosting agrega multiples árboles de decisión. La lógica de los algoritmos boosting es combinar modelos débiles para conseguir un modelo fuerte. En este caso, el término "débil" hace referencia a un modelo con poco poder predictivo pero mejor que una predicción aleatoria. Mientras que "fuerte" es considerado un buen predictor para los datos. Para llegar a un algoritmo fuerte a partir de varios modelos debiles es necesario introducir ponderaciones a los árboles basadas en las clasificaciones erroneas del árbol anterior.

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. El enfoque del boosting a apreder reiteradamente de los errores anteriores puede llevarle a generar sobreajuste. Aunque ese enfoque produce predicciones más precisas, muchas veces mejores a la mayoría de algoritmos, puede llevar a ajustar las observaciones atípicas. Por ello, el random forest es una técnica más recomendada para conjuntos de datos muy complejos con un gran número de observaciones atípicas.

Otra de las grandes desventajas del boosting es que tiene unos elevados tiempos de procesamiento dado que su entrenamiento sigue una lógica secuencial. Puesto que un árbol debe esperar al anterior para ser entrenado, se limita la escalabilidad del modelo. Sobre todo cuando se agregan más árboles. Mientras tanto, un random forest entrena los árboles en paralelo, lo que hace que sea más rápido de procesar. El inconveniente que es aplicable tanto a los algoritmos de boosting como a los de bagging y al random forest, es la perdidad de la simplicidad visual y la dificultad de interpretación que tienen respecto a un simple árbol de decisión.

[1]Universidad Complutense de Madrid

29.2. Gradient Boosting

Uno de los algoritmos de boosting más famosos es el gradient boosting. Mientras que el random forest seleccionaba combinaciones aleatorias de variables, el gradient boosting selecciona variables que mejoren la precisión con cada nuevo árbol. Por lo tanto, la construcción del modelo secuencial puesto que cada nuevo árbol se crea utilizando información derivada del árbol anterior, y en consecuencia no son independientes. En cada iteración se registran los errores cometidos en los datos de entrenamiento y se aplican a la siguiente ronda de datos de entrenamiento. Además, se agregan pesos a los datos basandose en los resultados de la iteración anterior. Las ponderaciones más altas se aplicarán a los datos que fueron errorneamente clasificados, y no se aplicará tanta atención a los bien clasificados. 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 árboles de decisión de regresión. Equivalentemente a la regresión logística, la predicción del ensamblado de los árboles de decisión es modelado utilizando una función sigmoidal tal que:

$$P(y = 1|x, f) = \frac{1}{1 + e^{-f(x)}} \tag{29.1}$$

donde $f(x) = \sum_{\kappa=1}^{K} f_\kappa(x)$ y f_m es un árbol de regresión. De nuevo, como en la regresión logística, se aplica el principio de máxima verosimilitud tratando de hayar una f que maximice $\mathcal{L}_f = \sum_{i=1}^{N} \ln(P(y_i = 1|x_i, f))$

El algorítmo en inicio es un modelo constante de la forma $f = f_0 = \frac{p}{1-p}$ donde $p = \frac{1}{N}\sum_{i=1}^{N}$. Tras cada iteración se añade un nuevo arbol f_κ al modelo. Para encontrar el mejor árbol f_κ, la primera derivada parcial g_i del modelo actual se obtiene para $i = 1, \ldots, N$:

$$g_i = \frac{\delta \mathcal{L}_f}{\delta f} \tag{29.2}$$

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. Notese que:

$$\ln(P(y_i = 1|x_i, f)) = \ln(\frac{1}{1 + e^{-f(x_i)}}) \tag{29.3}$$

y por tanto, la derivada respecto a f es igual a:

$$\frac{\delta \ln(P(y_i = 1|x_i, f))}{\delta f} = \frac{1}{e^{f(x_i)} + 1} \tag{29.4}$$

Figura 29.1: Ejemplo de Boosting

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 el paso de actualización óptimo ρ_κ como:

$$\rho_\kappa = \arg\max_{\rho} \mathcal{L}_{f+\rho f_\kappa} \tag{29.5}$$

Al terminar la iteración κ, se actualiza el modelo ensamblado f añadiendo el nuevo arbol f_κ:

$$f \leftarrow f + \alpha \rho_\kappa f_\kappa \tag{29.6}$$

Se itera hasta que $\kappa = K$, entonces el proceso se detiene y se obtiene el modelo ensamblado final f.

29.2.1. Hiperparámetros del modelo gradient boosting

Un modelo de gradient boosting tiene dos tipos de hiperparámetros:

1. Hiperparámetros de boosting
2. Hiperparámetros del árbol

29.2.1.1. Hiperparámetros de boosting

Los hiperparametos de boosting son principalmente dos: el número de árboles y la tasa de aprendizaje. El primero indica la cantidad de árboles que crecen de forma secuencial, y que 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. Además, el número de árboles necesarios por el GBM puede incrementarse debido a los valores que tomen 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 especificas 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 hablando y dificulta alcanzar el 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.

29.2.1.2. Hiperparámetros de árbol

Los principales hiperparametos de árbol son: la profundidad del árbol y el número mínimo de observaciones en nodos terminales. 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 hablando, pero necesitan más árboles. Sin embargo, los árboles de mayor profundidad permiten que el algoritmo capture interacciones únicas, aunque a su vez aumentan el riesgo de sobreajuste. El segundo hiperparámetro, además de controlar el número mínimo de observaciones en 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 un árbol, evitando así el sobreajuste. Sin embargo, los valores más pequeños pueden ayudar con clases objetivo desbalanceadas en problemas de clasificación.

29.2.2. Estrategia de ajuste de hiperparametros

A diferencia del random forest, los modelos gradient boosting pueden variar mucho en su precisión de acuerdo a su configuración de hiperparametros. 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 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 parámetros 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.

29.3. Procedimiento con R: la funcion `gbm`

En el paquete `gbm` de **R** se encuentra la función `gbm` que se utiliza para entrenar un modelo gradient boosting:

```
gbm(formula, data = ..., ...)
```

- 'formula': Que refleja el predictor lineal $Y \sim X_1 + ... + X_p$.
- 'data': Un conjunto de datos con el que interpretar el nombre de las variables incluidas en la formula.

29.4. Aplicación del modelo GBM 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 cuales no. Se contruye el modelo usando el conjunto de entrenamiento sin transformar (en su escala orginal). Así, en lugar de tener las variables categóricas transformadas a one-hot-encoding se guardan en su variable original, como ocurre con el caso de la variable que mide el nivel educativo.

```
library(CDR)
data(dp_entr)
```

```
# se fija una semilla común a todos los modelos
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_pro13 ~ .,
  data = dp_entr,
  method = "gbm",
  metric = "Accuracy",
  trControl = trainControl(
    classProbs = TRUE,
    method = "cv", number = 10
  )
)
```

```
#> 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 con 100 árboles, una profundidad de 3, ambos parametros han sido ajustados. Además, se habia dejado constantes tanto el número mínimo de observaciones en nodos igual a 10 y una tasa de aprendizaje de 0,1. Los resultados en el proceso de validación cruzada se muestra en la figura **??**, 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()
```

Figura 29.2: Resultados del modelo GB obtenidos durante el proceso de validación cruzada.

29.4.1. GBM con ajuste automático

Se repite el procedimiento para el ejemplo anterior. Sin embargo, en este ejemplo se va a proceder a ajustar de forma automática los hiperparámetros más relevantes de dicho algoritmo para mejorar los resultados respecto al modelo anterior. Se observa como los parámetros 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
#> 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 definimos rangos de valores para los principales hiperparámetros a optimizar.

```
# Se especifica un rango de valores típicos 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 parámetros del modelo se incorporan a la función de entrenamiento. Cuanto más exhaustiva sea la busqueda de estos valores, mayor será el tiempo de ajuste del modelo. La red presentada está formada por 81 combinaciones de los posibles cuatro hiperparametros.

```
# se fija una semilla común a todos los modelos
set.seed(101)


# se entrena el modelo
model <- train(CLS_PRO_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 hiperparametros a los siguientes valores: está compuesto de 180 árboles, con 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 el la figura **??** se ven los resultados obtenidos durante el proceso de validación cruzada. Se puede ver que los resultados son similares a los del modelo anterior, aunque hay puntos diferenciables 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()
```

Figura 29.3: Resultados del modelo GB con ajuste autómatico obtenidos durante el proceso de validación cruzada.

29.5. 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 solucionador eficiente de modelos lineales y un algoritmo de aprendizaje de árboles.

El paquete es compatible con funciones objetivo de regresión, clasificación y ranking. El paquete tiene varias características:

1. Velocidad: xgboost puede realizar automáticamente cálculos paralelos. Por lo general, es más de 10 veces más rápido que el modelo Gradient Boosting.
2. Tipo de entrada: xgboost toma varios tipos de datos de entrada:

- Matriz densa (matrix)
- Matriz dispersa (Matrix::dgCMatrix)
- Archivo de datos locales
- Un tipo de datos propio del paquete: xgb.DMatrix

3. Dispersión: xgboost acepta datos de entrada dispersos para los modelos incluidos.
4. Personalización: xgboost admite tanto funcion de objetivo y función de evaluación personalizadas.
5. Rendimiento: xgboost tiene un mejor rendimiento en varios conjuntos de datos.

29.5.1. Hiperparametros 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 parametros 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: gamma (γ), alpha (α) y lambda (λ). Gamma 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 gamma. 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 cumplan 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 proprociona una

regularización L_1, λ proprociona 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
```

29.6. Procedimiento con R: la funcion `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': Un conjunto de datos con el que entrenar el modelo.
- 'label': Vector con la variable respuesta.

29.7. Aplicación del módelo XGBoost en R

Este algoritmo vuelve a estar basado en árboles, así se contruye el modelo usando el conjunto de entrenamiento sin transformar (en su escala orginal). Se continua 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 parámetros por defecto que proporciona R.

```
# se fija una semilla común a todos los modelos
set.seed(101)
```

```
# se entrena el modelo
model <- train(CLS_PRO_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 (igual a 0) y para el tamaño mínimo del nodo (igual a 1). En cambio, ajusta los parametros del modelo dentro de los valores por defecto de la función. Así, el modelo XGBoost resultante ha sido uno con 50 iteraciónes, 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 parametros 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.

```
# resultados del modelo por cada iteración del cv
ggplot(melt(model$resample[, -4]), aes(x = variable, y = value, fill =
↪  variable)) +
  geom_boxplot()
```

Figura 29.4: Resultados del modelo XGBoost obtenidos durante el proceso de validación cruzada.

29.7.1. XGBoost y ajuste automático

Se continua el ejemplo de aplicar los datos sobre compra de un nuevo producto por parte de los clientes utilizando un modelo extreme gradient boosting en R. Sin embargo, se quieren mejorar los resultados obtenidos, y por ello se va a proceder a ajustar de forma automática los parámetros más relevantes de dicho algoritmo. Se genera una red de posibles valores para dichos parametros. Por motivos computacionales, esta no se hace muy exhaustiva para evitar largos tiempos de entrenamiento, pero es aconsejable tratar de estudiar más valores para los parámetros que interese 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 fija una semilla común a todos los modelos
set.seed(101)


# se entrena el modelo
model <- train(CLS_PRO_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, que la tasa de aprendizaje sea del 0,2 y que la regularización γ tome el valor 5.

Figura 29.5: Resultados del modelo XGBoost con ajuste autómatico obtenidos durante el proceso de validación cruzada.

Los resultados guardados 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 ha introducido al lector en el algoritmo de aprendizaje supervisado conocido como gradient boosting, en concreto:

- Se ha presentado 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 han expuesto los hiperparámetros más relevantes a la hora de optimizar un modelo de gradient boosting.
- Se ha presentado el extreme gradient boosting, una implementación eficiente y escalable del modelo gradient boosting. Así como los hiperparámetros de regularización y otros parametros importantes en esta implementación.
- Se han aplicado ambos algoritmos en `R` en un caso práctico para la clasificación binaria de datos.

Parte VI

Machine learning no supervisado

Capítulo 30

Análisis cluster: clusterización jerárquica

José-María Montero y Gema Fernández-Avilés

30.1. 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 los dos siguientes capítulos se centran en este último problema y nos referiremos a él por su denominación más popular: AC.

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, 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. 30.1).

```
library(ggplot2)
set.seed(1980)
```

```
x <- c(rnorm(1500, mean = -4), rnorm(1500, mean = 0))
y <- c(rnorm(1500, mean = 0), rnorm(1500, mean = 1.5))
cluster <- as.factor(c(rep("A", 1500), rep("B", 1500)))
xy <- data.frame(x, y, cluster)

ggplot(data = xy, aes(x = x, y = y)) +
  geom_point(aes(color = cluster)) +
  scale_color_manual(values = c("red", "blue"))
```

Figura 30.1: Datos simulados que presentan clusters

Para llevar a cabo un AC, se deben tomar una serie de decisiones:

- 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 los dos capítulos siguientes.

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.

```
library(CDR)
data("TIC2021")
```

30.2. Selección de las variables

La selección de las p variables o características, $\{X_1, X_2, ..., X_p\}$, 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 30.1: Información muestral

	X_1	X_2	X_3	$\cdots$	X_p
Elemento 1	x_{11}	x_{12}	x_{13}	$\cdots$	x_{1p}
Elemento 2	x_{21}	x_{22}	x_{23}	$\cdots$	x_{2p}
Elemento 3	x_{31}	x_{32}	x_{33}	$\cdots$	x_{3p}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Elemento n	x_{n1}	x_{n2}	x_{n3}	$\cdots$	x_{3p}

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 Capítulo 32, 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 Edelborck (1979) y Everitt (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.

[1] También 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 para no poder clasificar las variables que describen cada elemento (cluster por variables en vez de por elementos).

```
tic <- scale(TIC2021)
```

30.3. 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]

- $d(\mathbf{x}_i;\mathbf{x}_j) \geq 0$,
- $d(\mathbf{x}_i;\mathbf{x}_j) = 0$ 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 $\mathbf{x}_i = \mathbf{x}_j$,
- $s(\mathbf{x}_i;\mathbf{x}_j) = s(\mathbf{x}_j;\mathbf{x}_i)$,
- $|s(\mathbf{x}_i;\mathbf{x}_j) + s(\mathbf{x}_j;\mathbf{x}_k)|s(\mathbf{x}_i;\mathbf{x}_z) \geq d(\mathbf{x}_i;\mathbf{x}_k)s(\mathbf{x}_j;\mathbf{x}_k) \in \mathbb{R}^p$.

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:

$$d_e(\mathbf{x}_i;\mathbf{x}_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2} \tag{30.1}$$

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.

[2] Si se cumple el cuarto requisito la función distancia suele llamarse distancia métrica.

```
library(factoextra)
d_euclidea <- get_dist(x = tic, method = "euclidea")
as.matrix(d_euclidea)[1:5, 1:5]
#>          1        2        3        4        5
#> 1 0.000000 6.421631 2.417212 1.870962 2.304686
#> 2 6.421631 0.000000 4.616177 7.988106 4.871235
#> 3 2.417212 4.616177 0.000000 3.765714 1.366011
#> 4 1.870962 7.988106 3.765714 0.000000 3.607589
#> 5 2.304686 4.871235 1.366011 3.607589 0.000000
```

La Fig. 30.2 muestra un heatmap[3] de distancias euclideas entre los países de la UE27 a partir deen la Uni las estadísticas de uso de las TIC en 2021.

```
fviz_dist(dist.obj = d_euclidea, lab_size = 10)
```

Figura 30.2: Heatmap de distancias euclídeas: datos 'TIC2021' del paquete 'cdr'

- **Distancia Manhattan o city block.** Se define como:

$$d_{MAN}(\mathbf{x}_i;\mathbf{x}_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \,. \tag{30.2}$$

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.

[3]Representación visual de fácil lectura e interpretación basada en un código de colores, típica del analytics de páginas web. Normalmente proporcionan un patrón visual en forma de"F", que es el del seguimiento ocular de un sitio o plataforma tecnológica.

- **Distancia de Minkowski.** Se define como:

$$d_{MIN}(\mathbf{x}_i;\mathbf{x}_j) = \left(\sum_{k=1}^{p} |x_{ik} - x_{jk}|^{\lambda}\right)^{\frac{1}{\lambda}}. \tag{30.3}$$

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:

$$d_{CHE}(\mathbf{x}_i;\mathbf{x}_j) = \operatorname*{máx}_{1\leq k\leq p} \sum_{k=1}^{p} |x_{ik} - x_{jk}|. \tag{30.4}$$

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:

$$d_{MAH} = (\mathbf{x}_i;\mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)'\mathbf{S}^{-1}(\mathbf{x}_i - \mathbf{x}_j) \tag{30.5}$$

Coincide con la distancia euclídea calculada sobre las componentes principales. Es invariante a cambios de origen y de escala (por tanto, se puede sustituir $\mathbf{S}$ por $\mathbf{R}$. Además, tiene en cuenta, explícitamente, las correlaciones lineales que puedan existir entre las variables, corrigiendo así el efecto redundancia. Es, por tanto, es apropiada 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:

$$d_P(\mathbf{x}_i;\mathbf{x}_j) = \frac{\sum_{k=1}^{p}(x_{ik} - \overline{x}_i)(x_{jk} - \overline{x}_j)}{\sqrt{\sum_{k=1}^{p}(x_{ik} - \overline{x}_i)^2 \sum_{k=1}^{p}(x_{jk} - \overline{x}_j)^2}}. \tag{30.6}$$

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` (`euclidea`, '`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		
		Presencia	Ausencia	Total
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 Jacard**: $c_J = \frac{n_{11}}{n_{11}+n_{12}+n_{21}}$
- 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 disfutan de la relación de monotonicidad en el sentido anteriormente apuntado, siendo las 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 (capítulo 23.4).

Variables cualitativas (politómicas)

Cuando todas las variables sean cualitativas y alguna sea politómica, se generan para estas ultimas 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:

$$S_{ij}(\mathbf{x}_i;\mathbf{x}_j) = \frac{\sum_{k=1}^{p} s_{ij}}{\sum_{k=1}^{p} w_{ij}} \tag{30.7}$$

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 de ponderación; (ii) pasar las variables cuantitativas a cualitativas y utilizar las medidas propuestas para este tipo de variables.

30.4. Técnicas de agrupación jerárquicas

30.4.1. Introducción

Una vez se han seleccionado las variables en función de las cuales se van a agrupar en clusters o conglomerados los elementos disponibles, 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.

TÉCNICAS DE CLUSTERIZACIÓN:

1. **Jerárquicas**:
   - *Aglomerativas*:
     - Vecino más cerano 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 intragrupos
     - Método flexible de Lance y Williams
   - *Divisivas*:
     - Vecino más cerano 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 intragrupos
     - 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 puntos (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 particionan 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 a la que nos vamos a referir[4]. En lo que sigue, nos centramos en las técnicas jerárquicas, abordando las no jerárquicas en el Capítulo 31.

30.4.2. 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 tuviesen $n-1$ 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*".

[4] Elaboración propia en base a Kassambara (2017).
[5] El criterio de inclusión de los elementos en dichos conglomerados citado en 30.1.

Nótese que mientras que con el método del vecino más cerano 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 lejano tiende a separar a los individuos en menor medida que la indicada por sus disimilaridades iniciales (es espacio-contractivo), 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 disimilaridades iniciales (Gallardo-San Salvador and Vera-Vera, 2004).

- **Método de la distancia media.**

Surge como una solución a la constricción 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, un elemento de cada conglomerado. Igual que los dos métodos precedentes, es invariante a transformaciones monótonas de la distancia utilizada.

En la Fig. 30.3 se puede ver la constricción, 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.

```
library(factoextra)
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")

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")

library(patchwork)
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: $c_g = (\overline{x}_{1g}, \overline{x}_{2g}, ..., \overline{x}_{pg})$, donde $\overline{x}_{jg}$ es la media de la j-ésima variable en dicho grupo.

Figura 30.3: Clusterización jerárquica con distancias euclídeas (dendrograma): métodos del vecino más cercano, vecino más lejano y distancia media

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 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. 30.4, 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 corrije la limitación del 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.**

Figura 30.4: 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 euclidea 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. 30.5, 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 = "phylogenic"
)
```

- **Método del encadenamiento intragrupos.**

[6]Específicamente, 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 30.5: Clusterización jerárquica con distancias euclídeas al cuadrado (filograma): método de Ward

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 todos los del otro y computando, posteriormente, la media aritmética de dichas distancias. Con el método de la vinculación intragrupos 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 intergrupos entre los dos conglomerados es:

$$d_{intergrupos} = \frac{d_{(a;c)} + d_{(a;d)} + d_{(b;c)} + d_{(b;d)}}{4}$$

mientras que la distancia intragrupos vendrá dada por la media de las distancias entre los elementos a, b, c y d:

$$d_{intergrupos} = \frac{d_{(a;b)} + d_{(a;c)} + d_{(a;d)} + d_{(b;c)} + d_{(b;d)} + d_{(c;d)}}{6}$$

- **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:

$$d_{(g_1 \cup g_2);g_3} = \alpha_1 d_{(g_1;g_3)} + \alpha_2 d_{(g_2;g_3)} + \beta d_{(g_1;g_2)} + \gamma \left| d_{(g_1;g_2)} - d_{(g_2;g_2)} \right|,$$

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:

Tabla 30.3: 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

30.4.3. 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 subividir 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:

$$\mathbf{X} = \begin{pmatrix} . & . & . & . & . \\ 8 & . & . & . & . \\ 7 & 4 & . & . & . \\ 6 & 1 & 4 & . & . \\ 3 & 4 & 5 & 4 & . \end{pmatrix}$$

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 nuestro 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 2.2):

- (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 aquéllos elementos del grupo principal en los que $(i) - (ii) < 0$ se selecciona aquél para el cual es máxima. Tras esta segunda etapa los conglomerados son $\{e_1, e_5\}$ y $\{e_2, e_3, e_4\}$.

Tabla 30.4: 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 son positivas (en nuestro caso esto ocurre en la tercera etapa; véase Tabla 2.3).

Tabla 30.5: 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
$\{e_4\}$	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 nuestro caso, 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 2.4), y como todas las distancias son positivas, se divide $\{e_2, e_4\}$ en $\{e_2\}$ y $\{e_4\}$.

Tabla 30.6: 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 TIC, DIANA proporciona el dendrograma circular de la Fig. 30.6

```
# compute divisive hierarchical clustering
library(cluster)
hc_diana <- diana(tic, metric = "euclidea")
# Divise coefficient. #los valores más cercanos a 1 sugieren una estructura de
↪  agrupación fuerte
hc_diana$dc
#> [1] 0.8043393

library(factoextra)
fviz_dend(
  x = hc_diana,
  k = 3,
  type = "circular",
  ggtheme = theme_minimal()
)
```

Análisis de la asociación

En caso de que los elementos vengan caracterizados por variables cualitativas o factores dicotómicos, $F_1, F_2, ..., F_n$ (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:

Figura 30.6: Clusterización jerárquica divisiva con DIANA

- (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 restantes (véase Capítulo 23.1.2).

		Presencia	Factor j	
		SI	NO	**Total**
Presencia	SI	n_{11}	n_{21}	$n_{1\cdot}$
Factor i	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 Capítulo 23.2.5 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 con la finalidad 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 procede con un ANOVA entre las categorías de la variable independiente, que maximiza la varianza secuencial que se realiza mediante divisiones dicotómicas de la variable dependiente que busca en cada etapa la partición intergrupos y minimiza la varianza intragrupos. La agrupación de categorías se efectúa probando todas las combinaciones binarias posibles de las variables. Se utiliza un test F para seleccionar las mayores diferencias posibles. 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 difieran mucho en el número de categorías.
- Las particiones resultantes dependen de la variable que 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 variables 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.

30.5. Calidad de la agrupación y número de clusters

30.5.1. 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 triangulo 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 la magnitud de los correspondientes a otras técnicas de agregación es bastante parecida.

```
# comparamos con la distancia euclidea: 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_dia,
↪  cof_ward)

round(cor(coef_cofeneticos)[1, ], 2)
#> d_euclidea   cof_simp   cof_comp   cof_prom    cof_dia   cof_ward
#>       1.00       0.71       0.61       0.77       0.65       0.60
```

30.5.2. 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. 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 numero 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 abcisas 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. Otra posibilidad es utilizar el **gráfico de sedimentación** (32.4, que relaciona la variablidad 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, compútese la suma de cuadrados intragrupo (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 clúster y lejos de otros clústeres, mientras que -1 indica que están cerca de los clústeres vecinos.

El **criterio del Gap (brecha)**, similar al método del codo, tiene como finalidad encontrar la mayor diferencia o distancia que 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.

```
# Plot cluster results
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
) +
  ggtitle("Gap")

library(patchwork)
p1 + p2 + p3
```

Finalmente, el **índice de Dunn** es el cociente entre la mínima distancia intergrupos y la máxima distancia intragrupos. A mayor índice, mayor calidad de clusterización.

Figura 30.7: Métodos heurísticos para la determinación del número óptimo de clusters

```
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 nuestro 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 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 la contrastación 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 clústers.

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, estudiandose las no jerárquicas en el siguiente capítulo.

Capítulo 31

Análisis cluster: clusterización no jerárquica

José-María Montero y Gema Fernández-Avilés

Como se avanzó en 30.4.1, aunque las técnicas de agrupación jerárquicas son muy utilizadas, existen otras, también muy populares, que se aglutinan 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- o 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, y
- Los clusters resultantes no están anidados unos en otros.

31.1. 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,

las medidas 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.

31.1.1. 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 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-medias[1] 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 máximizan 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.

[1] Recuérdese que el centro de un conglomerado viene dado por el centroide, vector de medias.

En nuestro 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. 31.1).

```
library(factoextra)
set.seed(123)
kmeans_tic <- eclust(tic, "kmeans", k = 3)
```

Figura 31.1: Clusterización no jerárquica con k-medias

Algunas versiones del k-medias como el k-medias difusa, el k-medias recortadas, el k-medias armónicas, el k-medias sparse y el k-medias sparse robusto pueden verse en Carrasco-Oberto (2020).

31.1.2. Técnicas basadas en medoides

31.1.2.1. 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-medoids 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

[2]El 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.

(utiliza la distancia Manhattan) en lugar de la suma de los cuadrados de las distancias euclídeas[3]. Además, sus agrupaciones no dependen del orden en que han sido introducidos 1os elementos, cosa que puede ocurrir con otras técnicas no-jerárquicas, y 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 31.2: Clusterización no jerárquica con PAM

31.1.2.2. CLARA

La ineficiencia 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 muestreos. Solo una pequeña porción de los datos totales es seleccionada como representativa de los datos y los medoids son

[3] las 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.

[4] No 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.

escogidos de 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 (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 31.3: Clusterización no jerárquica con CLARA

31.1.2.3. 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).

31.1.3. 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 del las TIC es prácticamente la misma para técnicas jerárquicas aglomeratativas como el vecino más lejano, el método de Ward, el del centroide algoritmos divisivos como DIANA o las técnicas no jerárquicas con preselección de 3 grupos (k-medias, PAM y CLARA).

31.2. Métodos basados en la densidad de puntos

Utilizan indicadores de frecuencia, construyendo grupos mediante la detección de aquéllas 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 sigue una Normal si las variables son cuantitativas, y una 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. Pertenecen 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 escasísisma 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.

DBSCAN[5] es, quizás, el más popular. Incluso ha recibido premios por sus numerosísimas apli-

[5]La densidad se refiere al número de elementos en una misma zona. Sin embargo, si te paras a pensar, es un concepto subjetivo, porque, (ii) ¿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.

caciones a lo largo del tiempo. Soluciona los problemas de los métodos de reasignación, que son 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) y 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 ellaϵ-neighborhood[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 $\{Q_1 = P, Q_2, Q_3, ..., Q_n\}$ tal que Q_{i+1} es denso-alcanzable directamente desde Q_i, $\forall 1 \leq i \leq n$; (iii) Q está *denso-connectado* 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 u observación 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. Búsquense, recursivamente, todas las observaciones denso-conectadas con ella y asígnense al mismo grupo.
- Itérese el mismo proceso para todas todas las observaciones no visitadas.
- Aquellas observaciones que tras haber sido visitadas no pertenecen a ningún cluster se marcan como outliers.

Como resultado de 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 require 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

[6] Véase Amat Rodrigo (2017) para una discusión sobre el valor de ambos parámetros.

denso-alcanzables desde más de un cluster pueden asignarse a uno u otro dependiendo del orden 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 ha podido utilizar DBSCAN. Sin embargo, para ilustrar su utilización, en la Fig. 31.4 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 con cierto ruido (outliers); en consecuencia, los grupos no deberían ser esféricos y DBSCAN sería un algoritmo de agrupación adecuado.

Figura 31.4: Comparación entre los algorítmos k-means y DBSCAN para el conjunto de datos simulado 'multishapes'

31.3. 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 'biclust' proporciona varios algoritmos para encontrar clusters en dos dimensiones. Además, es muy recomendable para el preprocesamiento 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) haciéndo 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-means`, muy similar al `k-means` 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 $f(\mathbf{x}|\psi) = \sum_{g=1}^{G} \pi_g f_g(\mathbf{x}|\theta_\mathbf{g})$, donde π_g son las proporciones asignadas a cada grupo en la mixtura, tal que $\sum_{g=1}^{G} \pi_g = 1$; $f_g(x|\theta_g)$ es la función de densidad correspondiente al g-ésimo grupo y $\psi = (\pi_\mathbf{1}, \pi_\mathbf{2}, ..., \pi_\mathbf{G}, \theta_\mathbf{1}, \theta_\mathbf{2}, ...\theta_\mathbf{G})$. Las funciones de densidad $f_g(\mathbf{x}|\theta_\mathbf{g})$ 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-medias[7]. 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 considerar que las características de los elementos son independientes, y que no resulta adecuado 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). Una idea intuitiva puede encontrarse en Amat Rodrigo (2017).

31.4. Nota final

La elección de que técnica de clusterización, jerárquica o no, es una decisión del investigador, y dependerá de su deseo de clasificación, 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

[7] Esta es la razón para incluir este tipo de clusterización en entre las técnicas no jerárquicas.

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 posteriomente.

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 probabilistica (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 32

Análisis de componentes principales

José-María Montero y José Luis Alfaro Navarro

32.1. 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 2 a 2, 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 escalado mulidimensional (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 cluster (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 colinealidad); 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).[1] 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, entre otras, (*i*) incorpora notabes 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`.

32.2. Obtención de las componentes principales

32.2.1. Descripción formal del proceso

Sea $\mathbf{X}' = (X_1, \ldots, X_p)$ 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, $Y_j = a_{1j}X_1 + a_{2j}X_2 + \cdots + a_{pj}X_p = \mathbf{a}_j'\mathbf{X}, \quad j = 1, 2, \ldots, p$, denominadas *c.p.*, que recojan la variabilidad existente en los datos. La idea es ordenar las *c.p.* tal que $V(Y_1) > V(Y_2) > ... > V(Y_p)$, y seleccionar m de ellas (las m primeras), $m < p$, que capturen un elevado porcentaje de la variabilidad de los datos.[2]

La varianza de cada componente y la covarianza entre ellas vienen dadas por:

$$\begin{aligned} Var(Y_j) &= \mathbf{a}_j'\boldsymbol{\Sigma}\mathbf{a}_j, \quad \forall j = 1, 2, \ldots, p \\ Cov(Y_j, Y_k) &= \mathbf{a}_j'\boldsymbol{\Sigma}\mathbf{a}_k, \quad \forall j, k, j \neq k, = 1, 2, \ldots, p \end{aligned} \tag{32.1}$$

[1] Una descripción completa de la base de datos está disponible en https://geodacenter.github.io/data-and-lab/Guerry/

[2] Geométricamente, estas combinaciones lineales 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.

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 $Var(Y_1) = \mathbf{a}_1'\mathbf{\Sigma}\mathbf{a}_1$ sujeto a que $\mathbf{a}_1'\mathbf{a}_1 = 1$. En otros términos, se selecciona el vector $\mathbf{a}_1$ que maximiza el lagrangiano:

$$\mathcal{L}(\mathbf{a}_1) = \mathbf{a}_1'\mathbf{\Sigma}\mathbf{a}_1 - \lambda(\mathbf{a}_1'\mathbf{a}_1 - 1) \tag{32.2}$$

Para ello se deriva respecto a $\mathbf{a}_1$ y λ y se igualan a cero dichas derivadas:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{a}_1)}{\partial \mathbf{a}_1} = 2\mathbf{\Sigma}\mathbf{a}_1 - 2\lambda\mathbf{a}_1 = (\mathbf{\Sigma} - \lambda\mathbf{I})\mathbf{a}_1 = \mathbf{0} \\ \frac{\partial \mathcal{L}(\mathbf{a}_1)}{\partial \lambda} = \mathbf{a}_1'\mathbf{a}_1 - 1 = 0 \end{aligned} \tag{32.3}$$

La primera ecuación tendrá solución distinta del vector nulo cuando $(\mathbf{\Sigma} - \lambda\mathbf{I})$ sea singular. Es decir, cuando $|\mathbf{\Sigma} - \lambda\mathbf{I}| = 0$, o en otros términos, cuando λ sea un autovalor de $\mathbf{\Sigma}$. Dado que,

- $\mathbf{\Sigma}$ es semidefinida positiva y, en general, tendrá p autovalores no negativos;
- y que en el proceso de optimización, premultiplicando $(\mathbf{\Sigma} - \lambda\mathbf{I})\mathbf{a}_1 = \mathbf{0}$, por $\mathbf{a}_1'$ y teniendo en cuenta que $\mathbf{a}_1'\mathbf{a}_1 = 1$, resulta que $\lambda = \mathbf{a}_1'\mathbf{\Sigma}\mathbf{a}_1 = V(Y_1)$,[3]

se seleccionará el mayor de los autovalores de $\mathbf{\Sigma}$, 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 condicion de incorrelación con Y_1: $Cov(Y_1, Y_2) = \mathbf{a}_2'\mathbf{\Sigma}\mathbf{a}_1 = 0$, 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:

$$\mathcal{L}(\mathbf{a}_2) = \mathbf{a}_2'\mathbf{\Sigma}\mathbf{a}_2 - \lambda(\mathbf{a}_2'\mathbf{a}_2 - 1) - \gamma(\mathbf{a}_2'\mathbf{a}_1 - 0) \tag{32.4}$$

Derivando respecto a $\mathbf{a_2}$ e igualando a cero:

$$\frac{\partial \mathcal{L}(\mathbf{a}_2)}{\partial \mathbf{a}_2} = 2\mathbf{\Sigma}\mathbf{a}_2 - 2\lambda\mathbf{a}_2 - \gamma\mathbf{a}_1 = 2(\mathbf{\Sigma} - \lambda\mathbf{I})\mathbf{a}_2 - \gamma\mathbf{a}_1 = \mathbf{0} \tag{32.5}$$

Premultiplicando por $\mathbf{a}_1'$ y considerando la condición de ortogonalidad, se tiene que $\gamma = 2Cov(Y_1, Y_2) = 0$, con lo que $\frac{\partial \mathcal{L}(\mathbf{a}_2)}{\partial \mathbf{a}_2} = 2\mathbf{\Sigma}\mathbf{a}_2 - 2\lambda\mathbf{a}_2 = 0$, que implica que $(\mathbf{\Sigma} - \lambda\mathbf{I})\mathbf{a}_2 = 0$.

[3] La condición $\mathbf{a}_i'\mathbf{a}_i = 1$ lleva a que los autovalores de $\mathbf{\Sigma}$ coincidan con las varianzas de las *c.p.*

[4] Si todos los autovalores de $\mathbf{\Sigma}$ son distintos, los autovectores son ortogonales. En caso contrario, los autovectores asociados a autovalores comunes se eligen de forma que sean ortogonales.

Siguiendo el mismo razonamiento que en la obtención de la primera componente, se elige el segundo mayor autovalor de $\mathbf{\Sigma}$[5], λ_2, siendo $\mathbf{a_2}$ el autovector asociado a él.

Obtención del resto de las componentes principales

Repitiendo 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 $\mathbf{\Sigma}$.

El vector de componentes principales se puede expresar como $\mathbf{Y} = \mathbf{A'X}$, donde $\mathbf{A} = [\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_p}]$, es la matriz de autovectores (ortogonales) obtenidos.

32.2.2. Cuestiones importantes en el ACP

32.2.2.1. Varianza de las variables originales y las CP

La matriz de covarianzas de las *c.p*, $\mathbf{V}(\mathbf{Y}) = \mathbf{A'\Sigma A}$, coincide con $\mathbf{\Lambda}$, 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 $\mathbf{\Sigma}$. En consecuencia:

$$\sum_{i=1}^{p} Var(Y_i) = tr(\mathbf{\Lambda}) = tr(\mathbf{A'\Sigma A}) = tr(\mathbf{\Sigma A A'}) = tr(\mathbf{\Sigma}) = \sum_{i=1}^{p} Var(X_i), \tag{32.6}$$

pudiéndose comprobar que la suma de las varianzas de las variables originales[6] 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 la proporción capturada por las m primeras componentes $\frac{\sum_{j=1}^{m}\lambda_j}{\sum_{j=1}^{p}\lambda_j}100$.

32.2.2.2. CP a partir de variables estandarizadas

A menudo, no solo se centran las variables originales sino que también se estandarizan, para tener varianza unitaria. 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 $\mathbf{\Sigma}$; 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 $\mathbf{\Sigma}$ sino $\mathbf{P}$, la matriz de correlaciones de dichas variables. Los autovectores de $\mathbf{P}$ son, en general distintos a los de $\mathbf{\Sigma}$. Además, la suma de los autovalores, como coincide con la

[5] El 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.

[6] Recuérdese que se adopta como medida de la variabilidad de las variables originales la suma de sus varianzas.

suma de las varianzas de las variables originales, es p, luego el porcentaje de la variación total capturada por la componente j-ésima $\frac{\lambda_j}{p}100$, siendo la proporción capturada por las m primeras componentes $\frac{\sum_{j=1}^{m}\lambda_j}{p}100$.

32.2.2.3. Correlación entre las variables originales y las CP

Considérese la variable original X_i y la *c.p.* $Y_j = a_{1j}X_1 + a_{2j}X_2 + \cdots + a_{pj}X_p = \mathbf{a}_j'\mathbf{X}$. Dado que $\mathbf{X}' = [X_1, \cdots, X_p]$, 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 $(\mathbf{\Sigma} - \lambda_j\mathbf{I})\mathbf{a}_j = 0$, se tiene que $\mathbf{\Sigma}\mathbf{a}_j = \lambda_j\mathbf{a}_j$ y que

$$Cov(X_i, Y_j) = Cov(\mathbf{e}_i'\prime\mathbf{X}, \mathbf{a}_j'\mathbf{X}) = \mathbf{e}_i'\mathbf{\Sigma}\mathbf{a}_j = \mathbf{e}_i'\lambda_j\mathbf{a}_j = \lambda_j a_{ij} \tag{32.7}$$

$$r_{X_i,Y_j} = \frac{Cov(X_i, Y_j)}{\sqrt{Var(X_i)}\sqrt{Var(Y_j)}} = \frac{\lambda_j a_{ij}}{\sqrt{\sigma_{ii}}\sqrt{\lambda_j}} = \frac{\sqrt{\lambda_j}a_{ij}}{\sigma_{ii}} \tag{32.8}$$

donde σ_{ii} es el elemento i-ésimo de la diagonal principal de $\mathbf{\Sigma}$.

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 no varía por el hecho de haber estandarizado las variables originales.

Como se verá posteriormente, estos dos coeficientes 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 un elemento 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 en el eje de Y_j. Matricialmente, y denominando $\mathbf{A}^*$ a la matriz de coeficientes de correlación lineal entre las variabes originales tipificadas y las *c.p.*, se tiene que $\mathbf{A}^* = \mathbf{A}\mathbf{\Lambda}^{\frac{1}{2}}$. $\mathbf{A}^*$, 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 $\frac{r^2_{X_i,Y_j}}{\sum_{i=1}^{p} r^2_{X_i,Y_j}} = \frac{cos(X_i,Y_j)}{\sum_{i=1}^{p} cos(X_i,Y_j)}$.

Otras dos expresiones interesantes que involucran $\mathbf{A}^*$ son $\mathbf{A}^*\mathbf{A}^{*\prime} = \mathbf{R}$ y $\mathbf{A}^{*\prime}\mathbf{A}^* = \mathbf{\Lambda}$.

32.3. Estimación de las componentes principales

Hasta el momento, se han derivado las *c.p.* suponiendo conocida la matriz de covarianzas poblacional $\mathbf{\Sigma}$ (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 Fig. 32.1. 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")
library(corrplot)
corrplot(cor(TIC2021), method = "number", type = "lower")
```

Figura 32.1: Matriz de correlaciones

```
library(FactoMineR)
acp <- PCA(TIC2021, ncp = 7, graph = FALSE)
```

32.4. 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 la interpretación de las *c.p.* es muy difícil.

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 $\mathbf{\Sigma}$) o si $\lambda_j > 1$ (si se parte de $\mathbf{R}$). En caso de 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 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. 32.2 muestra el gráfico de sedimentación en el ejemplo que nos ocupa:

```
round(acp$eig[1:4, 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
library(factoextra)
fviz_eig(acp, addlables = TRUE)
```

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.

Figura 32.2: Gráfico de sedimentación

32.5. Interpretación de las CP

Una primera vía consiste en analizar el signo y la magnitud de los coeficientes (*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
```

- 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 32.3: Gráfico de cosenos o coeficientes de correlación variables-componentes

Se puede utilizar cualquiera de las dos, 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. 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.

Como puede apreciarse en la matriz de *loadings*, la primera *c.p.* es una media ponderada (con ponderaciones no muy distintas) de las variables originales, mientras que *esales*, *hbroad* y *hiacc* cargan fuertemente en la segunda, las primera positivamente y las dos segundas 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. La segunda *c.p.* está directamente relacionada con la dotación TIC de las empresas pero tiene una relación fuertemente negativa con la dotación TIC de los individuos y los hogares. **PUEDE INTERPRETARSE COMO XXXX**.

```
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 32.4: Contribución de las variables originales a las componentes retenidas

32.6. Reproducción de los datos tipificados y de R a partir de las CP

En la práctica, el punto de partida del ACP es la matriz $\mathbf{R}$, y en tal caso se suelen estandarizar también las *c.p.* Pues bien, se tiene que:

$$\mathbf{Y}^* = \mathbf{X}\mathbf{A}\mathbf{\Lambda}^{-\frac{1}{2}} = \mathbf{X}\mathbf{A}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{\Lambda}^{-1} = \mathbf{X}\mathbf{A}^*\mathbf{\Lambda}^{-1} = \mathbf{X}\mathbf{F},$$

donde $\mathbf{F}$ 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 nuestro caso:

```
puntuaciones <- acp$ind$coord
round(puntuaciones[1:5, ], 3)
#>    Dim.1  Dim.2  Dim.3  Dim.4  Dim.5  Dim.6  Dim.7
#> 1  1.651  1.053  0.310 -0.238 -0.196  0.296 -0.196
#> 2 -4.759 -0.127 -0.128 -0.223  0.023 -0.013 -0.106
#> 3 -0.324  0.875 -0.564  0.870 -0.082 -0.032 -0.345
#> 4  3.188  1.331  0.497  0.262  0.343 -0.300  0.319
#> 5  0.024  0.144 -0.183  0.560 -0.871 -0.485  0.115
```

De la expresión anterior se deduce que $\mathbf{X} = \mathbf{Y}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{A}' = \mathbf{Y}^*{}_{N\times m}\mathbf{A}^{*\prime}{}_{m\times p}$, que es la expresión que permite reproducir la matriz $\mathbf{X}$ a partir de las m primeras *c.p.* tipificadas. En consecuencia,

la reproducción de $\mathbf{R}$ a partir de las las m primeras *c.p.* tipificadas se lleva a cabo como sigue:

$$\begin{aligned}\mathbf{R}_{p\times p} = \frac{1}{N}\mathbf{X}'_{p\times N}\mathbf{X}_{N\times p} &= \mathbf{A}^*_{p\times m}\frac{1}{N}\mathbf{Y}^{*'}_{m\times N}\mathbf{Y}^*_{N\times m}\mathbf{A}^{*'}_{m\times p}\\ &= \mathbf{A}^*_{p\times m}\mathbf{I}_{m\times m}\mathbf{A}^{*'}_{m\times p} = \mathbf{A}^*_{p\times m}\mathbf{A}^{*'}_{m\times p}\end{aligned} \tag{32.9}$$

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. 32.1) se concluye que se trata de una buena reproducción.[7]

```
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
```

32.7. Limitaciones del ACP

Una primera limitación es que su implementación solo 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, solo tiene en cuenta 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.

El t-SNE no es la única alternativa no lineal procedente de la comunidad de machine learning. Otras, denominadas actualmente "aprendizaje múltiple" (manifold learning), son el Sammon's mapping, curvilinear component analysis (CCA) y sus variantes, Laplacian eigenmaps y maximum variance unfolding (MVU), entre otros; 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.*

[7] Con tres *c.p.* la reproducción es casi perfecta. Se han retenido solo dos para poder mostrar representaciones bidimensionales y para evitar la interpretación de una tercera componente.

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 33

Análisis factorial

José-María Montero y José Luis Alfaro Navarro

33.1. 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 observables[1]. En el marco lineal, y considerando p variables[2], 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 politómicas 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 opinasen 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 esté muy bien en el club pero quiera ir a otro más prestigio; o que esté mal, pero

[1] Se asumirá la representación lineal, por sencillez, pero puede ser cualquier otra.

[2] Por las mismas razones que en el análisis de componentes principales (ACP), se trabaja con las variables estandarizadas; véase Cap. 32.

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. 32) 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. 32 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")
```

33.2. Elementos teóricos del análisis factorial

33.2.1. Modelo básico y terminología

Considérense p variables $\{X_1, X_2, ..., X_p\}$ y N elementos, objetos o individuos, siendo las matrices de datos, $\mathbf{X}$, y datos estandarizados, $\mathbf{Z}$, las siguientes:

[3] No son los únicos.

[4] Ambos enfoques dan resultados similares cuando hay más de 30 variables y las comunalidades (véase Sec. 33.2.1) son superiores a 0,70, y se interpretan de manera casi idéntica.

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pN} \end{pmatrix}, \quad \mathbf{Z} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1N} \\ z_{21} & z_{22} & \cdots & z_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ z_{p1} & z_{p2} & \cdots & z_{pN} \end{pmatrix},$$

donde el primer subíndice indica la variable y el segundo el elemento.

Mientras que el enfoque de componentes principales está representado por:

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jp}F_p, \quad j = 1, 2, \cdots, p, \tag{33.1}$$

en el enfoque AF clásico el modelo teórico es:

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jk}F_k + b_jSP_j + c_jE_j, \quad j = 1, 2, \cdots, p, \tag{33.2}$$

donde $Z_j,\ j = 1, 2, \cdots, p$, se modeliza, linealmente, en términos de (i) $k \ll p$ factores comunes, $F_m,\ m = 1, 2, \cdots, k$, que dan cuenta de la correlaciones entre las variables $Z_j,\ j = 1, 2, \cdots, p$, y (ii) un factor específico, $SP_j,\ j = 1, 2, \cdots, p$, y un término de error, $E_j,\ j = 1, 2, \cdots, p$, que dan cuenta de la varianza no compartida (específica y residual, respectivamente). Los coeficientes a_{jm} se denominan $cargasfactoriales$ 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 aunarán el factor específico y el término de error de Z_j en un factor único, U_j.

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jk}F_k + d_jU_j, \quad j = 1, 2, \cdots, p, \tag{33.3}$$

Los supuestos del modelo (33.3) son los siguientes: (i) como en la práctica los factores comunes y únicos son desconocidos, sin pérdida de generalidad pueden suponerse con media cero y varianza unitaria; (ii) los factores únicos se suponen independientes entre sí y de los factores comunes; y (iii) dado que los factores involucrados en el modelo se consideran variables aleatorias, si se asume normalidad, e independencia de los factores comunes, $\{F_1, F_2, \cdots, F_k\}$ sigue una distribución normal multivariante y $Z_j,\ j = 1, 2, \cdots, p$, una distribución normal.

En términos de valores observados, el modelo AF (33.3) viene dado por:[5]

$$z_{ji} = a_{j1}f_{1i} + a_{j2}f_{2i} + \cdots + a_{jk}f_{ki} + d_ju_{ji}, \quad i = 1, 2, \cdots, N; j = 1, 2, \cdots, p, \tag{33.4}$$

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 solo pueden estimarse a partir de los datos observados. Los propios factores se estiman en una etapa posterior del análisis.

En términos matriciales, y considerando:

$$\mathbf{z} = \begin{pmatrix} Z1 \\ Z_2 \\ \vdots \\ Z_p \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_k \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_p \end{pmatrix},$$

[5] Aunque 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.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pk} \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_p \end{pmatrix},$$

el modelo (33.3) puede expresarse como $\mathbf{z} = \mathbf{Af} + \mathbf{Du}$.

Centrándonos en el modelo (33.3), la varianza de Z_j viene dada por:

$$V(Z_j) = 1 = \sum_{m=1}^{k} a_{jm}^2 + 2\sum_{m<q}^{k} a_{jm}a_{jq}r_{(F_{mi},F_{qi})} + d_j^2 \tag{33.5}$$

y si los factores comunes están incorrelados, $V(Z_j) = 1 = \sum_{m=1}^{k} a_{jm}^2 + d_j^2$.

De la expresión (33.5) 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 $Z_j,\ j = 1, 2, \cdots, p$.
- $V = \sum_{m=1}^{k} V_m$ es la contribuión de todos los factores comunes a la varianza de todas las variables $Z_j,\ j = 1, 2, \cdots, p$.
- $\frac{V}{p}$ es un indicador de la *completitud* del análisis factorial.
- $h_j^2 = a_{j1}^2 + a_{j1}^2 + \cdots + a_{jk}^2$ es la *comunalidad* de $Z_j,\ j = 1, 2, \cdots, p$, es decir la contribución de los factores comunes a la variabilidad de Z_j.
- d_j^2 es la unicidad de $Z_j,\ j = 1, 2, \cdots, p$, 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 (33.2)), b_j^2 se denomina *especificidad* 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".

33.2.2. Patrón y estructura factoriales

Se denomina *patrón factorial* a la siguiente expansión del modelo (33.2),

$$\begin{aligned} Z_1 &= a_{11}F_1 + a_{12}F_2 + \cdots + a_{1k}F_k + d_1U_1 \\ Z_2 &= a_{21}F_1 + a_{22}F_2 + \cdots + a_{2k}F_k + d_2U_2 \\ &\vdots \qquad \vdots \qquad \ddots \qquad \vdots \quad \vdots \\ Z_p &= a_{p1}F_1 + a_{p2}F_2 + \cdots + a_{pk}F_k + d_pU_p \end{aligned} \tag{33.6}$$

o simplemente a la tabla, o matriz, con los coeficientes a_{jm} y d_j (o únicamente, caso habitual, a la matriz de cargas $\mathbf{A}$). k determina la "complejidad del modelo".

Se denomina *estructura factorial* al siguiente conglomerado de p conjuntos de $k+1$ ecuaciones lineales en $\{a_{jm}\}$, las p primeras, y en $\{d_j\}$, la última, $j = 1, 2, \cdots, p;\ m = 1, 2, \cdots, k$:[6]

$$
\begin{aligned}
r_{Z_jF_1} &= a_{j1}r_{F_1F_1} + a_{j2}r_{F_1F_2} + \cdots + a_{jm}r_{F_1F_m} + \cdots + a_{jk}r_{F_1F_k} \\
&\vdots \\
r_{Z_jF_m} &= a_{j1}r_{F_mF_1} + a_{j2}r_{F_mF_2} + \cdots + a_{jm}r_{F_mF_m} + \cdots + a_{jk}r_{F_mF_k} \\
&\vdots \\
r_{Z_jF_k} &= a_{j1}r_{F_kF_1} + a_{j2}r_{F_kF_2} + \cdots + a_{jm}r_{F_kF_m} + \cdots + a_{jk}r_{F_kF_k} \\
&\vdots \\
r_{Z_jU_j} &= d_j \\
&\vdots
\end{aligned}
\tag{33.7}
$$

En la práctica, viene dada por una tabla, o matriz, $\mathbf{\Gamma}$, 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

$$
\mathbf{F} = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \cdots & f_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{k1} & f_{k2} & \cdots & f_{kN} \end{pmatrix}, \quad \mathbf{\Gamma} = \begin{pmatrix} r_{Z_1F_1} & r_{Z_1F_2} & \cdots & r_{Z_1F_k} \\ r_{Z_2F_1} & r_{Z_2F_2} & \cdots & r_{Z_2F_k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{Z_pF_1} & r_{X_p^*Z_pF_2} & \cdots & r_{Z_pF_k} \end{pmatrix},
$$

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 $\mathbf{\Phi}$ 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

[6] En caso de variables dicotómicas se utiliza el coeficiente ϕ (véase 23.4) como medida de correlación momento-producto.

[7] La historia es más larga: el AF es indeterminado porque dada una matriz $\mathbf{C}_{kxk}$ no singular, si se define otro vector de factores comunes $\mathbf{f}^* = \mathbf{C^{-1}f}$ y otra matriz $\mathbf{A}^* = \mathbf{AC}$, entonces $\mathbf{Z} = \mathbf{Af} + \mathbf{Du} = \mathbf{A}^*\mathbf{C^{-1}Cf}^* + \mathbf{Du} = \mathbf{A}^*\mathbf{f}^* + \mathbf{Du}$ y ambos son equivalentes. Una solución es exigir la incorrelación de los factores comunes ($\mathbf{\Phi} = \mathbf{I}$), con lo que la indeterminación se reduciría solo a cuando $\mathbf{C}$ sea ortogonal. En este caso, la solución será única salvo rotaciones ortogonales.

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]

33.3. El análisis factorial en la práctica

33.3.1. Pre-análisis factorial

33.3.1.1. ¿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. Contrasta la incorrelación de las variables, es decir, $H_0 : \mathbf{R} = \mathbf{I}$ frente a $H_1 : \mathbf{R} \neq \mathbf{I}$ (o $H_0 : |\mathbf{R}| = \mathbf{1}$ frente a $H_1 : |\mathbf{R}| = \mathbf{1}$). El estadístico de contraste es $d_{\mathbf{R}} = -\left(N - 1 - \frac{1}{6}(2p+5)\right) ln|\mathbf{R}|$ 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
```

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

[8] Una solución determina el espacio k-dimensional que contiene los k factores comunes pero no su posición exacta.

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 $KMO = \dfrac{\sum_j \sum_{i \neq j} r_{ji}^2}{\sum_j \sum_{i \neq j} r_{ji}^2 + \sum_j \sum_{i \neq j} r_{ji}^{*2}}$, 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 elevadísismas correlaciones en $\mathbf{R}$; 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: $MSA_j = \dfrac{\sum_{i \neq j} r_{ji}^2}{\sum_{i \neq j} r_{ji}^2 + \sum_{i \neq j} r_{ji}^{*2}}$.

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 comunalidades 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.

33.3.1.2. El problema de la comunalidad 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 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 comunalidades (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 comunalidades, o se hipotetiza sobre dicho rango o se hipotetizan o estiman las comunalidades. 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: Núm. factores comunes $= k$.
- (2) Se estiman las comunalidades y se obtienen los factores comunes.

En cuanto prefijar un número k de factores, se pueden seguir los criterios expuestos en Cap. 32 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 comunalidades, 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 comunalidad de una variable por su correlación más alta con las demás variables: $\hat{h}_j^2 = max(r_{j1}, r_{j2}, \cdots, r_{j(j-1)}, r_{j(j+1)}, \cdots, r_{jp})$.
- 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 comunalidad 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 comunalidades: $\hat{h}_j^2 \geq r^2_{Z_j;(Z_{j1},\cdots,Z_2,\cdots Z_{j-1},Z_{j+1},\cdots,Z_p)} = 1 - \frac{1}{r^{jj}}$, donde r^{jj} es el j-ésimo elemento de la diagonal de $\mathbf{R}^{-1}$.

Un valor alto de la comunalidad, próximo a $V(X_j)$, significa que dicha variable está bien representada en el espacio de factores.

33.3.2. Análisis factorial

33.3.2.1. Métodos de extracción de los factores

Método de las componentes principales

Su objetivo es el análisis de toda la varianza, común y no común, (modelo (33.1)). Por consiguiente, las entradas de la diagonal de $\mathbf{R}$[9] son unitarias y no se requiere la estimación a priori

[9] Cuando se utiliza este método se debe utilizar $\mathbf{R}$ porque, de otra forma, los factores comunes no tendrían media cero y varianza unitaria.

de las comunalidades; tampoco 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`, nos remitimos al Cap. 32. Aunque en Cap. 32 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 (33.3) puede dar estimaciones de las cargas factoriales muy sesgadas, sobre todo cuando hay variables con comunalidades 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 comunalidades en la diagonal en vez de unos. Exige, por tanto, la estimación previa de las comunalidades y su objetivo es el análisis de la varianza compartida por todas las variables (modelo (33.3)). 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 comunalidades utilizando el coeficiente de determinación lineal múltiple de cada variable con las demás.[10]

3.- Cálculo de la matriz de correlaciones reducida:

$$\mathbf{R}^* = \begin{pmatrix} \hat{h}^2_{1(0)} & r_{12} & \cdots & r_{1p} \\ r_{21} & \hat{h}^2_{2(0)} & \cdots & r_{2p} \\ \vdots & \vdots & \cdots & \vdots \\ r_{p1} & r_{p2} & \cdots & \hat{h}^2_{p(0)} \end{pmatrix}$$

.

4.- Cálculo de los autovalores y autovectores asociados a $\mathbf{R}^*$ (matriz no necesariamente semidefinida positiva) y a partir de ellos obtener 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.

5.- A partir de la estimación de $\mathbf{A_{(0)}}$, obtención de una nueva estimación de las comunalidades: $\hat{h}^2_{j(1)} = \hat{a}^2_{j1(1)} + \hat{a}^2_{j2(1)} + \cdots + \hat{a}^2_{jk(1)}$ y, por tanto, una nueva estimación de la varianza específica $\hat{\psi}^2_{j(1)} = 1 - \hat{h}^2_{j(1)}$.

6.- Comparación de $\hat{h}^2_{j(1)}$ con $\hat{h}^2_{j(0)}$, $j = 1, 2, \cdots, p$. 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 R, el método de los factores principales se implementa con la función `fa` de la librería `psych`, que parte de $\mathbf{R}^*$ (véase Harman (1976) para el procedimiento iterativo y Revelle (2022) para los detalles sobre la manera cómo `fa` parte de $\mathbf{R}^*$ y lleva a cabo la extracción de los factores).

[10] Una estimación inicial de las comunalidades equivale a una estimación inicial de las varianzas específicas: $\hat{d}^2_{j(0)} = \hat{\sigma}^2_j - \hat{h}^2_{j(0)}$; y si las variables originales están tipificadas: $\hat{d}^2_{j(0)} = 1 - \hat{h}^2_{j(0)}$.

```
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, *SSloadings* son los autovalores de $\mathbf{R}^*$, 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 comunalidades (suma de las cargas al cuadrado por filas; solo 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 es el número de factores comunes necesarios para explicar la variable Z_j: $c_j = \frac{\left(\sum_{j=1}^{p} a_{jm}^2\right)^2}{\sum_{j=1}^{p} a_{jm}^4}$. Cuanto mayor es *com* 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 comunalidades y especificidades 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. 33.3.2.2).

3.- **Método de Máxima Verosimilitud**

Exige normalidad multivariante y la determinación a priori del número de factores comunes, pero no la estimación de las comunalidades. Obedece al modelo (33.3) 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 $\mathbf{A}'(\mathbf{DD}')^{-\mathbf{1}}\mathbf{A}$ sea diagonal. La log-verosimilitud viene dada por $l = -\frac{N}{2}\left(log|2\pi\mathbf{\Sigma}| + tr\mathbf{\Sigma}^{-1}\mathbf{S}\right)$, donde $\mathbf{\Sigma} = \mathbf{AA}' + \mathbf{DD}'$ y $\mathbf{S}$ son las matrices de covarianzas poblacional y muestral, respectivamente, de las variables X_j, $j = 1, 2, \cdots, p$. $\mathbf{DD}'$ 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 $\mathbf{\Psi}$.

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. 33.3.3). 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=ml`). Otra posibilidad es utilizar la función `factanal`. En ambos casos hay comprobar el cumplimiento de la hipótesis de normalidad. En nuestro ejemplo `TIC` no procede su implementación al no cumplirse tal hipótesis.

Otros métodos

Razones de espacio impiden comentar otros métodos de extracción de los factores. Pero, al menos, queremos señalar que la función `fa` de la librería `psych` también permite implementar los métodos (i) minres (mínimo residuo), que estima las cargas 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 comunalidades, que se obtienen como subproducto, tras la estimación de las cargas; y (ii) alpha, que maximiza el alfa de Cronbach para los factores. Aunque `fa` también proporciona el método del centroide o la descomposición triangular (que exigen la estimación de las comunalidades), 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 observadas y reproducidas, en el último caso ponderando los coeficientes de correlación inversamente a la unicidad de las variables (alta unicidad supone

baja comunalidad). Ambos son proporcionados por `fa` y `FAiR`, que también es una librería muy recomendable.

33.3.2.2. 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 dificultosa, 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 en el espacio de las cargas correspondientes a X_j. Al realizar la rotación se tienen 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 (33.3) puede escribirse como $\mathbf{Z} = \mathbf{A}\mathbf{T}\mathbf{T}'\mathbf{F} + \mathbf{U} = \mathbf{B}\mathbf{T}'\mathbf{F} + \mathbf{U}$. 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 oblícuas, dependiendo de si los nuevos factores siguen estando incorrelacionados (ejes perpendiculares) o no (ejes oblicuos).

33.3.2.2.1. Rotaciones ortogonales Preservan la perpendicularidad de los ejes y no varían las comunalidades, pues $\mathbf{B}\mathbf{B}' = \mathbf{A}\mathbf{T}\mathbf{T}'\mathbf{A}' = \mathbf{A}\mathbf{A}'$. Tampoco modifican los cuadrados de las comunalidades 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}, $j = 1, 2, \cdots, p$: $SMPL_m = \frac{\sum_{j=1}^{p} b_{jm}^4}{p} - \left(\frac{\sum_{j=1}^{p} b_{jm}^2}{p}\right)^2$. 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, el método 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 comunalidad, y por tanto con mayores cargas factoriales, tendrán mayor influencia en la solución final, porque la comunalidad no se ve afectada por la rotación ortogonal. Para evitar esto, Kaiser propuso la rotación VARIMAX NORMALIZADA[12], donde las cargas se dividen entre la raíz cuadrada de la comunalidad 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 comunalidades, $h_j = \sum_{m=1}^{k} b_{jm}^2$, el segundo término de la expresión anterior no se verá modificado, por lo que el criterio anterior equivale a maximizar $\frac{\sum_{j=1}^{p}\sum_{m=1}^{k} b_{jm}^4}{pk}$.

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.

[11] Este 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 elementos de la fila, elevados al cuadrado, es la comunalidad fija de la variable.

[12] La normalización Kaiser se aplica también en los demás métodos.

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).

33.3.2.2.2. 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 oblícuo (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, $m = 1, 2, \cdots, k$: para cada factor rotado se puede encontrar un nuevo factor incorrelado con los rotados. A esos nuevos factores de 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, $j = 1, 2, \cdots, p$ 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 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 $\mathbf{\Lambda}$.
- *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 $K = \frac{\sum_{j=1}^{p}\sum_{m=1}^{k} v_{jm}^4}{\left(\sum_{j=1}^{p}\sum_{m=1}^{k} v_{jm}^2\right)^2}$. 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 el método BIQUARTIMIN, que consiste en minimizar $B^* = H + \frac{C^*}{p}$, donde $\frac{C^*}{p}$ es la expresión completa del COVARIMIN. Una generalización del 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 $\gamma = \frac{\beta}{\alpha+\beta}$. El criterio QUARTIMIN se obtiene con $\gamma = 0$, el BIQUARTIMIN con $\gamma = 0,5$ y el 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 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 oblícuas, pero únicamente se mencionarán (*i*) el método ORTOBLICUO, que llega a la solución oblicua mediante una serie de transformaciones ortogonales intermedias; y (*ii*) el método 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.

33.3.2.2.3. ¿Rotaciones ortogonales u oblícuas? 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 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 libería `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`.

33.3.3. 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 el 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 $\mathbf{R}^{rep} = \mathbf{A}\mathbf{\Phi}\mathbf{A}' = \mathbf{\Gamma}\mathbf{A}'$ (relación fundamental entre el patrón y la estructura factorial; en caso de incorrelación entre los factores, $\mathbf{\Phi} = \mathbf{I}$ y $\mathbf{R}^{rep} = \mathbf{A}\mathbf{A}'$. $\mathbf{R}^{rep}$ 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 $\sigma_{r=0} = \frac{1}{\sqrt{N-1}}$, $S_{r_{res}} \leq \frac{1}{\sqrt{N-1}}$:[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 $S_{r_{res}} \leq pero\ no \ll \frac{1}{\sqrt{N-1}}$ la solución es aceptable.

Otra posibilidad, también muy sencilla, propuesta por Revelle (2022), es $fit = 1 - \frac{\sum(\mathbf{R}-\mathbf{FF}')^2}{\sum(\mathbf{R})^2}$, 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 RMS error (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:

```
round(af_facprin2$residual, 3)
#>           ebroad esales esocmedia   eweb hbroad  hiacc   iuse
#> ebroad     0.505 -0.068     0.008  0.068 -0.045  0.002  0.003
#> esales    -0.068  0.447     0.026  0.015  0.004 -0.012  0.021
#> esocmedia  0.008  0.026     0.322 -0.047  0.014 -0.005  0.017
#> eweb       0.068  0.015    -0.047  0.182  0.012  0.015 -0.042
#> hbroad    -0.045  0.004     0.014  0.012  0.131 -0.005  0.010
#> hiacc      0.002 -0.012    -0.005  0.015 -0.005  0.018  0.002
#> iuse       0.003  0.021     0.017 -0.042  0.010  0.002  0.125
af_facprin2$rms
#> [1] 0.02907475
af_facprin2$fit
#> [1] 0.9715865
```

[13] Este criterio tiene como ventaja la simplicidad. Sin embargo, sería conveniente que tuviese en cuenta, al menos, el número de variables.

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 (33.2)(con los factores comunes y el factor único). Pues bien, si en el proceso reproductivo se utiliza el modelo solo con los factores comunes, la matriz de correlaciones que se reproduce es $\mathbf{R}$, lo que implica el modelo ACP (modelo (33.1)). 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 comunalidades.

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 $-2ln\lambda = np(\hat{a} - ln\hat{g} - 1]$, donde $\hat{a}$ y $\hat{g}$ son las medias aritmética y geométrica, respectivamente, de los autovalores de la matriz $\hat{\mathbf{\Sigma}}_{H_0}^{-1}\mathbf{S}$. Bajo H_0, se distribuye asintóticamente como una χ^2_{df}, 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 continua hasta que no se rechace la hipótesis nula, siempre que el valor de k sea compatible con un número de grados de libertad positivo.

33.3.4. Puntuaciones factoriales.

Las puntuaciones factoriales son las estimaciones de los valores de los factores aleatorios no observados, es decir, de los elementos de $\mathbf{F}_{mxm}$. 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 en los que se trabaje con los mismos elementos o individuos sustituyendo las variables originales por los nuevos factores obtenidos (regresión, cluster, etc.). 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 (i) el de regresión por mínimos cuadrados ordinarios (MCO), donde $\hat{\mathbf{F}} = (\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}'\mathbf{Z}$; (ii) el

[14] df indica la medida en que el modelo factorial ofrece una interpretación más simple que $\mathbf{\Sigma}$.

de Bartlett, basado en el método de estimación por mínimos cuadrados generalizados (MCG), con $\hat{\mathbf{F}} = \left(\mathbf{A}'\mathbf{\Psi}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}'\mathbf{\Psi}^{-1}\mathbf{Z}$. El mismo estimador se puede obtener por máxima verosimilitud asumiendo normalidad multivariante; (*iii*) el de Thompson (con un enfoque bayesiano), donde $\hat{\mathbf{F}} = \left(\mathbf{I} + \mathbf{A}'\mathbf{\Psi}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}'\mathbf{\Psi}^{-1}\mathbf{Z}$; y (*iv*) el de Anderson-Rubin (que obtiene estimaciones MCG imponiendo la condición $\mathbf{F}'\mathbf{F} = \mathbf{I}$ ($\hat{\mathbf{F}} = \left(\mathbf{A}'\mathbf{\Psi}^{-1}\mathbf{R}\mathbf{\Psi}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}'\mathbf{\Psi}^{-1}\mathbf{Z}$). Las ventajas y desventajas de cada uno de ellos pueden verse en Mardia et al. (1979a) y De la Fuente (2011).

En nuestro ejemplo de las TIC, las puntuaciones de los dos factores extraídos con el método de 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
#> [1,]  0.6256359  1.01289866
#> [2,] -2.1820404 -0.03439974
#> [3,] -0.2189723  1.08635525
```

33.4. 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 correlación. Sin embargo, AF asume que los datos proceden de un modelo bien definido, el modelo (33.3), 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, sobrestima 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 33.3.2.1, 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 en estudio (no de toda, 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 34

Escalamiento multidimensional

José María Montero y José Luis Alfaro Navarro

34.1. 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 multivariables 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) utilizando la información proporcionada por las distancias entre los datos. Esta técnica surgue 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,.... estudiados.

El EMD analiza matrices de proximidad (similitud, disimililitud 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, pudiendose describir las relaciones entre los objetos sobre la base de las proximidades observadas (López-Gónzalez and Hidalgo-Sánchez, 2010)

En R, existen distintas funciones para desarrollar el EMD, desde las clásicas funciones cmdscale de la librería **base** e isoMDS del paquete **MASS** hasta el enfoque más actual, usado en este documento, recogido en la librería **smacof** (de Leeuw and Mair, 2009; Mair et al., 2022) que proporciona al usuario una gran flexibilidad para especificar EMD. Utiliza siempre matrices de disimilaridad 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.

```
library(smacof)
library(CDR)
data("TIC2021")
```

34.2. 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 "disimilaridad", 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 disimilaridad) 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:

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}, 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 díficil 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:

$$S = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{pmatrix}$$

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 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:

a) La distancia euclídea :

$$d_e(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

b) La distancia absoluta, o city-block :

$$d_1(x, y) = \sum_{i=1}^{p} |x_i - y_i|$$

c) La distancia de Mahalanobis, usando la matriz de covarianzas de las variables:

$$d_M^2(x, y) = (x - y)'\mathbf{\Sigma}^{-1}(x - y)$$

Cuando las variables son binarias (0 y 1), los coeficientes de similaridad más utilizados son:

a) Coeficiente de Jaccard :

$$s_{ij} = \frac{a}{a+b+c}$$

b) Coeficiente de Sokal-Michener :

$$s_{ij} = \frac{a+d}{a+b+c+d}$$

donde a, b, c y d son las frecuencias de (1,1), (1,0), (0,1) y (1,1)c, respectivamente.

Por último, en el caso más general en el que existan variables cuantitativas, binarias y/o cualitativas, se suele utilizar la distancia de Gower, $d_{ij}^2 = 1 - s_{ij}$, donde:

$$s_{ij} = \frac{\sum_{h=1}^{p_1} (1 - |x_{ih} - x_{jh}|G_h) + a + \alpha}{p_1 + (p_2 - d) + p_3}$$

siendo p_1 el número de variables cuantitativas y G_h el rango de la h-ésima variable; a y d son el número de coincidencias y no coincidencias para las p_2 variables binarias; y α el número de coincidencias para las p_3 variables cualitativas.

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.

34.3. Modelo escalamiento multidimensional.

El EMD parte de una matriz de proximidades entre n objetos:

$$\mathbf{\Delta}_{nxn} = \begin{pmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & \cdots & \delta_{nn} \end{pmatrix}$$

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:

$$\mathbf{X}_{nxm} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

de forma que se puede calcular la distancia euclídea entre cada par de objetos

$$d_{ij} = \sqrt{\sum_{t=1}^{m} (x_{it} - x_{jt})^2}$$

y, construir una matriz de distancias "reproducidas"

$$\mathbf{D}_{nxn} = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}$$

que aproxime, en la medida de lo posible, a la matriz de proximidades, $\mathbf{\Delta}$.

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:

$$Stress = \sqrt{\frac{\sum (f(\delta_{ij}) - d_{ij})^2}{\sum d_{ij}^2}}$$

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:

$$S - Stress = \sqrt{\frac{\sum (f(\delta_{ij})^2 - d_{ij}^2)^2}{\sum (d_{ij}^2)^2}}$$

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:

$$RSQ = \frac{[\sum (d_{ij} - d_{..})(f(\delta_{ij}) - f(\delta_{..}))]^2}{[\sum (d_{ij} - d_{..})^2][\sum (f(\delta_{ij}) - f(\delta_{..}))^2]}$$

Este valor oscila entre 0 y 1, y se intrepreta de forma contraria a las medidas de Stress: mientras mayor sea el RSQ, mejor ajuste del modelo.

34.4. 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, mientras que en el segundo se considera que los datos están en escala ordinal.

34.4.1. 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 exitente entre observaciones, en este caso los países europeso.

En el primer caso, los "objetos" son las siete variables 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.

```
correlacion <- cor(TIC2021)
correlacion[1:3, ]
#>              ebroad    esales esocmedia      eweb    hbroad     hiacc
↪  iuse
#> ebroad    1.0000000 0.3768434 0.5874489 0.7043619 0.4223139 0.5211193
↪  0.6320885
#> esales    0.3768434 1.0000000 0.5420147 0.5845905 0.1666307 0.1945647
↪  0.4789584
#> esocmedia 0.5874489 0.5420147 1.0000000 0.6979517 0.5669977 0.6085944
↪  0.7563839
```

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 similidades (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
```

El gráfico con la representación de los objetos según sus distancias "reproducidas" muestra la configuración final de los siete objetos:

```
plot(res)
```

La información numérica detallada se podria obtener con la información de la salidad 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$stress" para obtener la medida de stress de Krukal y "res$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 que compara las proximidades originales y las distancias obtenidas:

```
plot(res, plot.type = "Shepard")
```

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:

```
res2 <- mds(datos, type = "interval")
plot(res2, plot.type = "Shepard")
```

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 correlaciónes, por lo tanto:

```
library(factoextra)
d_euclidea <- get_dist(x = TIC2021, method = "euclidea")
res <- mds(d_euclidea, ndim = 2, type = "ratio")
res
#>
#> 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(res)
```

En este caso, 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 Lituanía, Croacia, Irlanda y la República Checa.

34.4.2. 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.

Continuando con el ejemplo anterior, 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 debe utilizar el método "ordinal" dentro de la función mds:

```
res3 <- mds(datos, type = "ordinal")
res3
#>
#> Call:
#> mds(delta = datos, type = "ordinal")
#>
#> Model: Symmetric SMACOF
#> Number of objects: 7
#> Stress-1 value: 0.021
#> Number of iterations: 65
```

La medida de stress desciende hasta 0,021 y el gráfico de Shepard muestra una alta concordancia entre las ordenaciones original y reproducida. La información númerica detallada se puede obtener de forma similar al caso métrico.

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 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 librerías 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 35

Análisis de correspondencias

Román Mínguez Salido y *Manuel Vargas Vargas*

35.1. 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 de los niveles de los factores. Es decir, es un método de reducción de la dimensionalidad para factores representables en pocas dimensiones.

El punto de partida es una **tabla de contingencia RxC** T (véase capítulo anterior **??**) que recoge la frecuencia de cada par de niveles $A_1, A_2, ..., A_R$ del factor A y $B_1, B_2, ..., B_C$ del factor B:

	B_1	B_2	...	B_C	Total
A_1	n_{11}	n_{12}	...	n_{1C}	$n_{1\mathring{u}}$
A_2	n_{21}	n_{22}	...	n_{2C}	$n_{2\mathring{u}}$
...	...	...	...	...	...
A_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R\mathring{u}}$
Total	$n_{\mathring{u}1}$	$n_{\mathring{u}2}$	...	$n_{\mathring{u}C}$	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.

Si los factores fueran independientes, el **valor esperado** en cada casilla sería $e_ij = \frac{n_{i\mathring{u}} n_{\mathring{u}j}}{N}$, 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", $R = (r_{ij})$ 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 la referencias Greenacre (2008) (en español) o Beh and Lombardo (2014). En el resto del capítulo se hará una breve exposición de la metodología y se ejemplificará con el análisis de una tabla de contingencia.

35.2. 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, $d_{ii'} = \sum_{k=1}^{C} \frac{1}{n_{\text{ů}k}} \left(\frac{n_{ik}}{n_{i\text{ů}}} - \frac{n_{i'k}}{n_{i'\text{ů}}} \right)^2$
- para los perfiles columna, $d_{jj'} = \sum_{k=1}^{R} \frac{1}{n_{k\text{ů}}} \left(\frac{n_{kj}}{n_{\text{ů}j}} - \frac{n_{kj'}}{n_{\text{ů}j'}} \right)^2$

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

$$A = \begin{pmatrix} a_1' \\ \vdots \\ a_R' \end{pmatrix}, \text{ con } a_i = (a_{i1}\ a_{i2})', \text{ y } \ B = \begin{pmatrix} b_1' \\ \vdots \\ b_C' \end{pmatrix}, \text{ con } b_j = (b_{j1}\ b_{j2})' \tag{35.1}$$

siendo a_i las coordenadas del nivel fila A_i y b_j las del nivel columna B_j en el plano, de forma que:

- *"reproduzcan"* las distancias entre perfiles fila y columna y las diferencias tipificadas (asociaciones):

$$\begin{aligned} d(a_i, a_{i'}) &= \sqrt{(a_{i1} - a_{i'1})^2 + (a_{i2} - a_{i'2})^2} \approx d_{ii'} \\ d(b_j, b_{j'}) &= \sqrt{(b_{j1} - b_{j'1})^2 + (b_{j2} - b_{j'2})^2} \approx d_{jj'} \\ a_i' * b_j &\approx r_{ij} \end{aligned} \tag{35.2}$$

Con las coordenadas contenidas en las matrices A y B, es posible *"visualizar"* la posición relativa de cada factor. 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 $R = (r_{ij})$ en valores singulares. Según la importancia que se de al ajuste de uno de los perfiles o a la matriz de residuos, se obtienen diferentes métodos de selección, llamados **normalizaciones**, que pueden consultarse en Greenacre (2008).

35.2.1. Proyecciones fila, columna y simétrica

El punto de partida es la matriz de frecuencias relativas $P = T(n_{ij})/N$, también llamada **matriz de correspondencias**. Definiendo el vector de unos $\mathbf{1}$, con las dimensiones adecuadas, las masas 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 $r = P\mathbf{1}$ y $c = P'\mathbf{1}$ o, en forma de matrices diagonales,

$$D_R = diag(r) \equiv diag(f_{1\mathring{u}}, ..., f_{R\mathring{u}}) \text{ y } \; D_C = diag(c) \equiv diag(f_{\mathring{u}1}, ..., f_{\mathring{u}C})$$

Se calcula la matriz de **residuos estandarizados** como

$$S = D_R^{-\frac{1}{2}}(P - rc')D_C^{-\frac{1}{2}} \tag{35.3}$$

Esta matriz S se descompone en valores singulares, calculando matrices U, D y V tales que:

$$\begin{array}{c} S = UDV' \\ UU' = V'V = I \;\; , \;\; U_{(RxK)} \; V_{(CxK)} \; K = min(R-1, \; C-1) \\ D = diag(\mu_1, ..., \mu_K) \end{array} \tag{35.4}$$

donde los μ_i son los llamados **valores singulares**, estando ordenados de forma decreciente $\mu_1 \geq \mu_2 \geq ... \geq \mu_K$.

A partir de esta descomposición se pueden obtener:

- las **coordenadas estándares de las filas** $\Phi = D_R^{-\frac{1}{2}}U$, y sus **coordenadas principales** $F = \Phi D$.
- las **coordenadas estándares de las columnas** $\Gamma = D_C^{-\frac{1}{2}}V$, y sus **coordenadas principales** $G = \Gamma D$.
- 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 $A = D_R^{-\frac{1}{2}}UD$ y $B = D_C^{-\frac{1}{2}}VD$ representan las coordenadas de ambos perfiles en un espacio común, llamado **mapa simétrico**.

35.3. Procedimiento con R: la función `ca()`

Para realizar un análisis de correspondencias simple con **R** se puede utilizar la librería **ca**, que contiene la función `ca()`. Esta función acepta como argumento de entrada o bien directamente una tabla de contingencia, así como los datos originales como objeto matriz o data-frame. Incluso, el argumento puede ser una fórmula del tipo ~ F1 + F2 donde F1 y F2 son factores. Entre los argumentos adicionales se pueden incluir el número de dimensiones en el output así como la posibilidad de incluir filas o columnas suplementarias.

Como primer ejemplo, se van a utilizar los datos *"housetasks"*, contenidos en la librería **factoextra** y 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 permite comprobar el grado de asociación entre ambos factores:

```
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 aplicación de la función `ca()` proporciona los valores singulares y, tanto para filas como para columnas, las masas, las distancias chi-cuadrado, y las inercias explicadas, así como las coordenadas en el espacio proyectado:

```
ca_house <- ca(housetasks, k = 2)
ca_house
#>
#>  Principal inertias (eigenvalues):
#>            1        2        3
#> Value      0.542889 0.445003 0.127048
#> Percentage 48.69%   39.91%   11.4%
#>
#>
#>  Rows:
#>           Laundry Main_meal    Dinner Breakfeast   Tidying    Dishes
↪  Shopping
#> Mass     0.100917  0.087729  0.061927   0.080275  0.069954  0.064794
↪  0.068807
#> ChiDist  1.152997  1.016747  0.785880   0.715740  0.593815  0.549821
↪  0.466440
#> Inertia  0.134160  0.090692  0.038246   0.041124  0.024667  0.019587
↪  0.014970
#> Dim. 1  -1.346122 -1.188346 -0.939962  -0.690273 -0.534477 -0.256462
↪  -0.159717
```

```
#> Dim. 2  -0.742517 -0.734702 -0.461866  -0.678779  0.651108  0.662533
↪  0.604596
#>          Official   Driving Finances Insurance   Repairs Holidays
#> Mass     0.055046  0.079702 0.064794  0.079702  0.094610 0.091743
#> ChiDist  0.984014  1.128542 0.675490  0.852589  1.818512 1.462801
#> Inertia  0.053300  0.101509 0.029564  0.057936  0.312874 0.196311
#> Dim. 1   0.307586  1.006731 0.367485  0.878213  2.074861 0.342675
#> Dim. 2  -0.380181 -0.979506 0.926221  0.710229 -1.295584 2.151159
#>
#>
#>  Columns:
#>              Wife Alternating   Husband  Jointly
#> Mass     0.344037    0.145642  0.218463 0.291858
#> ChiDist  0.935393    0.899443  1.321252 1.038436
#> Inertia  0.301019    0.117824  0.381373 0.314725
#> Dim. 1  -1.136821   -0.084397  1.575600 0.202801
#> Dim. 2  -0.547487   -0.437116 -0.902313 1.538902
```

Las dos primeras dimensiones explican el 48.69 % y 39.91 %, por lo que la representación en un plano engloba al 88.6 % de la inercia. Los valores *"Mass"* corresponden a las masas (frecuencias relativas) de filas y columnas respectivamente.

Las distancias chi-cuadrado 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). Así, estos valores 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 distantes respecto del centroide de columnas es *"Husband"* (1.321).

Las inercias representan la distancia cuadrática χ^2 respecto al perfil promedio (sin calcular raíces), ponderada por la masa (de la fila o columna) correspondiente. Como la inercia mide la variabilidad, 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 (map= "rowprincipal", what=c("all","none")) o de las columnas (mapa "colprincipal", what=c("none","all")), tal como se muestra en la fig. 35.1:

```
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")
```

Respecto a las filas, se muestran varios grupos: el compuesto por *"Breakfast"*, *"Dinner"*, *"Main_meal"* y *"Laundry"*; otro por *"Shopping"*, *"Dishes"* y *"Tidying"*; uno tercero por

Figura 35.1: Proyecciones de los perfiles fila y columna

"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 (mapa "symmetric", what=c("all","all")), como se muestra en la fig. 35.2.

```
par(mfrow = c(1, 1))
plot(ca_house, map = "symmetric", what = c("all", "all"), xlab = "Proyección
↪  común de ambos factores")
```

Figura 35.2: 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"*.

Como segundo ejemplo, se van a utilizar los datos *"accidentes2020_data"*, contenidos en la librería **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"*, *"Despañamiento"* y *"Otro"* del factor *"tipo_accidente"* y los niveles *"Granizando"*, *"Nevando"*, *"NULL"* y *"Se desconoce"* del factor *"estado_meteorológico"*.

```
library(CDR)
# library(dplyr)
library(dtplyr)
data("accidentes2020_data")
datos <- as.data.frame(cbind(accidentes2020_data$tipo_accidente,
↪  accidentes2020_data$estado_meteorológico))
datos_depu <- filter(datos, V1 != "Atropello a animal")
datos_depu <- filter(datos_depu, V1 != "Atropello a persona")
datos_depu <- filter(datos_depu, V1 != "Caída")
datos_depu <- filter(datos_depu, V1 != "Solo salida de la vía")
datos_depu <- filter(datos_depu, V1 != "Vuelco")
datos_depu <- filter(datos_depu, V1 != "Despeñamiento")
datos_depu <- filter(datos_depu, V1 != "Otro")
datos_depu <- filter(datos_depu, V2 != "Granizando")
datos_depu <- filter(datos_depu, V2 != "Nevando")
datos_depu <- filter(datos_depu, V2 != "NULL")
datos_depu <- filter(datos_depu, V2 != "Se desconoce")
```

```
table(datos_depu)
#>                                V2
#> V1                              Despejado Lluvia débil LLuvia intensa
↪  Nublado
#>   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 = 103.99, df = 15, p-value = 2.275e-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     0.268637                     0.159370         0.033512
#> ChiDist  0.032004                     0.081686         0.065908
#> Inertia  0.000275                     0.001063         0.000146
#> Dim. 1  -0.160159                    -1.281759         0.729688
#> Dim. 2   1.365544                    -0.920696        -1.847953
#>         Colisión fronto-lateral Colisión lateral Colisión múltiple
#> Mass                   0.303605         0.154547          0.080329
#> ChiDist                0.044584         0.076822          0.128386
#> Inertia                0.000603         0.000912          0.001324
#> Dim. 1                 0.658572         1.229958         -2.081267
#> Dim. 2                -0.822076         0.528562          0.121030
#>
#>
#>  Columns:
#>         Despejado Lluvia débil LLuvia intensa  Nublado
#> Mass     0.861212     0.062076       0.010062 0.066650
#> ChiDist  0.013443     0.214467       0.289545 0.083914
#> Inertia  0.000156     0.002855       0.000844 0.000469
#> Dim. 1   0.205524    -3.461906      -3.671159 1.122905
#> Dim. 2  -0.190070    -0.838079       8.057832 2.020069
```

Las dos primeras dimensiones explican el 87.97 % y 11.4 %, por lo que la representación en un plano engloba al 99.37 % de la inercia.

Las representación gráfica de la proyección simétrica se muestra en la fig. 35.3:

```
plot(ca_tabla, map = "symmetric", what = c("all", "all"), xlab = "Proyección
↪  común de ambos factores")
```

Figura 35.3: Proyección simétrica de ambos factores

Se observa que *"Lluvia intensa"* no está especialmente asociada a ningún tipo de accidente; *"Lluvia débil"* con *"Colisión múltiple"* y *"Choque con obstáculo fijo"*; *"Despejado"* con *"Colisión fronto-lateral"*, *"Colisión frontal"* y *"Alcance"*; y *"Nublado"* con *"Colisión lateral"*.

RESUMEN
El análisis de correspondencias permite, dada una tabla de contingencia, reproducir las distancias entre niveles de cada factor en un espacio de menor dimensión, permitiendo la comparación gráfica entre ellos, así como la representación en un espacio común de los niveles de ambos factores.
En el primer caso, permite una visualización de la composición interna de cada factor, identificando los niveles que más se distancien del centroide. En el segundo caso, permite la representación de la asociación entre niveles de cada uno de los factores.

Parte VII

Deep learning

Capítulo 36

Redes neuronales artificiales

Noelia Vállez Enano y José Luis Espinosa Aranda

36.1. ¿Qué es el *deep learning*?

La inteligencia artificial es el conjunto de técnicas que hacen que cualquier elemento controlado por un computador sea capaz de realizar acciones similares a las que haría un humano en situaciones determinadas. Entre otros ejemplos podemos 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. 36.1). 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).

36.1.1. Diferencias entre las técnicas de *machine learning* tradicional y el *deep learning*

Como se vio en capítulos anteriores, las técnicas de *machine learning* tradicional requieren al inicio realizar una selección de las mejores características que representen el problema a resolver, y que puedan ser comprendidas por el algoritmo seleccionado de tal forma que sea capaz de solucionar el problema planteado.

Figura 36.1: Inteligencia Artificial vs Machine learning vs Deep Learning

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. 36.2). Estas características proporcionarían la base para detectar a medio nivel elementos de la cara como ojos, narices, orejas, etc. y, definitivamente a alto nivel, reconocer donde hay una cara dentro de la imagen.

Figura 36.2: 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 y algoritmos que permita resolver el problema planteado.

Debido a esto, nacen las técnicas de *deep learning*, las cuales tratan de simular el comportamiento de aprendizaje humano. Esto implica que, 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.

Por ejemplo, continuando con el ejemplo anterior de la detección de caras, mientras que en las técnicas de *machine learning* sería necesario explicarle al algoritmo que 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 si mismo.

La 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 iniciales y una mayor potencia de cómputo.

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 ha podido empezar a utilizar estas técnicas gracias a:

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 al incorporar una gran cantidad de microprocesadores específicos, inicialmente, para representar modelos complejos 3D en los monitores, pero que han podido ser utilizadas para las técnicas de *deep learning*, llevando recientemente al desarrollo de tarjetas específicas para este fin .

2. **Big data.** La gran cantidad de datos que se generan y almacenan en la actualidad en el día a día, así como la mayor facilidad a la hora de trabajar con esos conjuntos de datos, han permitido cubrir la necesidad de datos iniciales necesarios.

3. **La evolución del sofware.** 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, lo cual ha llevado a la creación y mejora de diversos frameworks y aplicaciones relacionadas con el entrenamiento y despliegue de redes neuronales. Entre ellos, serían destacables Keras, Tensorflow, Pytorch, Caffe2, Matlab y OpenVINO.

36.2. 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 lleva años presente, 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 que clase, perteneciente al conjunto de clases utilizado para entrenar, pertenece una determinada imagen.

2. **Detección de objetos.** Permite localizar los objetos contenidos en una imagen y su tipología marcándose con un rectángulo. 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. 36.3).

Figura 36.3: 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 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 aún cuando se encuentren situadas de forma contigua.

4. **Reconocimiento del habla.** Permite a un computador procesar el habla humana en formato escrito. En la actualidad existen varios asistentes inteligentes basados en esta tecnología 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. 36.4).

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 dicho texto. Un ejemplo de este último caso sería Dall-E (Borji, 2022) (Fig. 36.5).

Figura 36.4: Traductor automático basado en Deep Learning

Figura 36.5: 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 viajar 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. 36.3).

36.3. Redes Neuronales

Las redes neuronales artificiales (en inglés Artificial Neural Network (ANN)) tienen su origen a finales de los años 50 a partir del 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.

36.4. Perceptrón o Neurona

El elemento básico de toda ANN es el perceptrón o neurona. Se trata de un modelo artificial basado 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, $\boldsymbol{w}$ 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 36.6).

Para cada vector de entrada, $\boldsymbol{x}$, la neurona aplicará los pesos, $\boldsymbol{w}$, como el producto escalar de ambos vectores:

$$\boldsymbol{w}'\boldsymbol{x} = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n. \tag{36.1}$$

Una vez obtenida la suma ponderada, se puede separar las entradas en dos conjuntos, obteniéndose como salida final un valor binario, siguiendo la fórmula:

Figura 36.6: Estructura del perceptrón o neurona

$$f(\boldsymbol{w}'\boldsymbol{x}) = \begin{cases} 1 & \text{si } \boldsymbol{w}'\boldsymbol{x} > 0 \\ 0 & \text{en otro caso} \end{cases}. \tag{36.2}$$

36.4.1. Aprendizaje

El proceso de aprendizaje del perceptrón busca el ajuste automático de los valores de los pesos. Estos pesos 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:

$$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}, \tag{36.3}$$

donde cada muestra, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, pertenece a una de las 2 clases, $y_j = \{0, 1\}$.

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 $\boldsymbol{x}_i$ del conjunto de datos:

$$\hat{y}_i(t) = f(\boldsymbol{w}(t)^T \boldsymbol{x}_i) = f(w_0(t) + w_1(t) \cdot x_{i1} + \cdots + w_n(t) \cdot x_{in}). \tag{36.4}$$

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:

$$w_j(t+1) = w_j(t) + \lambda \cdot |y_i - \hat{y}_i(t)| \cdot x_{ij}. \tag{36.5}$$

donde $|y_i - \hat{y}_i(t)|$ será 0 cuando la clase predicha coincida 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 r es 0.

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.

36.4.2. 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 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.

36.5. Perceptrón Multiclase

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:

$$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}, \tag{36.6}$$

donde cada muestra $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ pertenezca a una de las c clases posibles:

$$y_j = \{0, 1, \ldots, c-1\}. \tag{36.7}$$

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:

$$F = \{f_0, f_1, \ldots, f_{c-1}\} f_j : \mathbb{R}^n \to \mathbb{R}. \tag{36.8}$$

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":

$$\hat{y}_i = argmax_j(f_j(\boldsymbol{x}_i))j \in \{0, 1, \ldots, c-1\}. \tag{36.9}$$

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:

$$[(f_1(\boldsymbol{x}_i)), (f_2(\boldsymbol{x}_i)), \ldots, (f_c(\boldsymbol{x}_i))]. \tag{36.10}$$

36.6. 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.

$$f(x) = x \tag{36.11}$$

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.

$$f(x) = \begin{cases} 0 & \text{si } x < u \\ 1 & \text{en otro caso} \end{cases} \tag{36.12}$$

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")
```

- **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.

$$f(x) = \frac{1}{1 - e^{-x}} \tag{36.13}$$

Representándose gráficamente de la siguiente forma:

```
f <- function(x) {
  1 / (1 + exp(-x))
}
plot_activation_function(f, "Sigmoide", 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 sigmoide.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{36.14}$$

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 redes neuronales (Nair and Hinton, 2010).

$$f(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{en otro caso} \end{cases} \tag{36.15}$$

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))
```

36.7. Perceptrón Multicapa

Aunque el perceptrón puede representar muchos tipos de lógica, no es posible realizar con él la operación XOR (o exclusiva) 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.

Un MLP está compuesto por varias capas con neuronas. La primera capa será la capa de entrada, que recibirá la variables que representan los elementos del problema a resolver. Por otro lado, la última capa será, de forma similar a cada neurona individual, 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 36.7). Este tipo de capas también son llamadas *densas* o *totalmente conectadas*.

36.7.1. 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 $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ donde cada $\boldsymbol{x}_i \in \mathbb{R}^d$ e $y_i \in \{0, 1\}$, la salida de la primera capa, $\boldsymbol{z}_1$, para una entrada $\boldsymbol{x}$ vendrá dada por la expresión:

$$\boldsymbol{z}_1 = \boldsymbol{W}'_{(1)}\boldsymbol{x} + \boldsymbol{b}_1, \tag{36.16}$$

Figura 36.7: Estructura del perceptrón multicapa (MLP)

donde $\boldsymbol{b}_1 \in \mathbb{R}^h$ es un vector con las constantes de la primera capa y $\boldsymbol{W}_{(1)} \in \mathbb{R}^{h \times d}$ son los pesos de la capa. Tras aplicar la función de activación, $g(\cdot)$, al vector intermedio, $\boldsymbol{z} \in \mathbb{R}^h$, se obtiene:

$$\boldsymbol{h_1} = g(\mathbf{z_1}). \tag{36.17}$$

La salida de una capa intermedia, $\boldsymbol{h_i} \in \mathbb{R}^h$, 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:

$$\boldsymbol{h}_2 = g(\boldsymbol{W}'_{(2)}\boldsymbol{h}_1 + \boldsymbol{b}_2). \tag{36.18}$$

Siguiendo el mismo razonamiento, la salida de la última capa, $\hat{y}$, y por tanto de la red, vendrá dada por:

$$\hat{y} = g(\boldsymbol{W}'_{(n)}\boldsymbol{h}_{n-1} + \boldsymbol{b}_n). \tag{36.19}$$

Por ejemplo, si se tiene una red de tres capas la salida podrá calcularse como:

$$\hat{y} = g(\boldsymbol{W}'_{(3)}\boldsymbol{g}(\boldsymbol{W}'_{(2)}\boldsymbol{g}(\boldsymbol{W}'_{(1)}\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3). \tag{36.20}$$

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

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 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;

$$C(\hat{y}, y, \boldsymbol{W}) = -\frac{1}{n}\sum_{i=0}^{n}(y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i)) + \frac{\alpha}{2n}||\boldsymbol{W}||_2^2, \tag{36.21}$$

donde $\alpha||W||_2^2$ 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:

$$C(\hat{y}, y, \boldsymbol{W}) = \frac{1}{2n}\sum_{i=0}^{n}||\hat{y}_i - y_i||_2^2 + \frac{\alpha}{2n}||\boldsymbol{W}||_2^2. \tag{36.22}$$

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, $\nabla C_{\boldsymbol{W}}$ se calcula y deduce de $\boldsymbol{W}$. Formalmente esto puede expresarse como:

$$\boldsymbol{W}^{t+1} = \boldsymbol{W}' - \lambda \nabla C_{\boldsymbol{W}}^{t}, \tag{36.23}$$

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 de él llamado *batch*. El conjunto de entrenamiento se divide, por tanto, en un número de *batches* 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.

36.8. 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("Hellow Tensorflow")
```

```
tf.Tensor(b'Hellow Tensorflow', shape=(), dtype=string)
```

36.9. Ejemplo de red para clasificación

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 36.8).

Figura 36.8: MLP para reconocimiento de dígitos manuscritos

36.9.1. 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 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 (dígito contenido en la imagen). 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 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 36.9: Algunas imágenes del conjunto de entrenamiento

36.9.2. 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
```

36.9.3. 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
```

[1] La activación *softmax* convierte un vector de número reales en una distribución de probabilidad.

```
    optimizer = "sgd", # stochastic gradient descent
    metrics = "accuracy" # Precisión
  )
```

36.9.4. 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 *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 36.10: Evolución durante el entrenamiento de la precisión y la pérdida de los conjuntos de entrenamiento y validación

Como se puede observar, la red entrenada tiene alrededor de un 90 % de precisión para las imágenes en los conjuntos de entrenamiento y validación.

36.9.5. 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:

```
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 qué pares de clases está confundiendo:

```
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. Estos resultados coinciden con el valor de **accuracy** calculado mediante la función **evaluate** previa.

36.9.6. 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
```

36.10. Ejemplo de red para regresión

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 inmobiliaros del año 2018 y que fue utilizado en el Capítulo @ref(#chap-feature). 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.

36.10.1. 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)
```

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, ]
```

36.10.2. 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. Para ello, se utilizará la función *scale* en las variables predictoras y se dividirá la variable del precio entre 100000 para reducir su orden:

```
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
```

36.10.3. Generación de la red neuronal

El siguiente paso consiste en la generación de la red neuronal. Para ello, al igual que en la sección 36.9.3, 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 activación debido a que el valor de la misma ya es comprensible tanto para el modelo como para su interpretación, y cualquier función de activación restringiría el rango de valores que podría obtener.

```
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
  )
```

36.10.4. 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 36.9.4:

```
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)
```

Figura 36.11: 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.

36.10.5. 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 37

Redes neuronales convolucionales

Noelia Vállez Enano y José Luis Espinosa Aranda

37.1. Introducción

Las redes neuronales convolucionales (del 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. 37.1).

Figura 37.1: 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" en los últimos años (Goodfellow et al., 2016).

Tras las capas convolucionales suelen encontrarse las capas *densas* o *totalmente conectadas* que no son otra cosa que una red neuronal artificial tal y como vimos en el capítulo 36 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 y una parte de clasificación (o regresión).

37.2. Convolución

Aunque las redes neuronales artificiales pueden utilizarse con los valores de color de una imagen como variables para reconocer qué hay en ella (ver capítulo 36), 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 37.2 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, $\boldsymbol{M}$, 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 37.2: Ejemplo de convolución. De izquierda a derecha: entrada, filtro y salida.

En general, una convolución en dos dimensiones se define como:

$$\boldsymbol{M}[x,y] = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \boldsymbol{F}[s,t]\boldsymbol{I}[x-s, y-t], \tag{37.1}$$

donde $\boldsymbol{F}$ es el filtro a aplicar, $\boldsymbol{I}$ es la matriz de entrada, $\boldsymbol{M}$ 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 37.2 se obtiene como:

$$\begin{aligned}
M_{1,1} &= 1\cdot 0+0\cdot 0+1\cdot 0+0\cdot 0+1\cdot 1+0\cdot 1+1\cdot 0+0\cdot 1+1\cdot 1=2\\
M_{1,2} &= 1\cdot 0+0\cdot 0+1\cdot 0+0\cdot 1+1\cdot 1+0\cdot 1+1\cdot 1+0\cdot 1+1\cdot 1=3\\
&\cdots\\
M_{2,2} &= 1\cdot 1+0\cdot 1+1\cdot 1+0\cdot 1+1\cdot 1+0\cdot 1+1\cdot 1+0\cdot 1+1\cdot 1=5\\
&\cdots\\
M_{3,3} &= 1\cdot 1+0\cdot 1+1\cdot 0+0\cdot 1+1\cdot 1+0\cdot 0+1\cdot 0+0\cdot 0+1\cdot 0=2
\end{aligned} \tag{37.2}$$

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 37.3 muestra el resultado de aplicar distintos filtros a una imagen de entrada.

Figura 37.3: Resultado de aplicar diferentes filtros de convolución sobre una imagen dada.

Los valores (o pesos) de los filtros, que se ajustaban tradicionalmente de forma manual según el problema a resolver, 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.

37.3. Neuronas convolucionales

Las capas convolucionales de la CNN no estarán compuestas por perceptrones, sino por neuronas convolucionales que realizan dicha operación. 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 $\boldsymbol{Y}_j$ se calcula como la combinación lineal de las salidas de las neuronas de la capa anterior $\boldsymbol{Y}i$ operando cada una de ellas con el filtro $\boldsymbol{F}_{ij}$ correspondiente a esa conexión de forma que:

$$\boldsymbol{Y}_j = g(\boldsymbol{B}_j + \sum \boldsymbol{F}_{ij} \otimes + \boldsymbol{Y}_i), \tag{37.3}$$

donde $\boldsymbol{B}_j$ 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. 37.4)

Figura 37.4: 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 37.5 muestra algunos de los rellenos más empleados.

37.4. 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

Figura 37.5: Resultado de aplicar un filtro 3D a una imagen antes y después de pasar por el filtro de activación

valor, con el valor más cercano del borde, etc. La Figura 37.6 muestra algunos de los rellenos más empleados.

3	6	8
1	9	4
5	2	7

0	0	0	0	0
0	3	6	8	0
0	1	9	4	0
0	5	2	7	0
0	0	0	0	0

Valor Constante

3	3	6	8	8
3	3	6	8	8
1	1	9	4	4
5	5	2	7	7
5	5	2	7	7

Valores del borde

9	1	9	4	9
6	3	6	8	6
9	1	9	4	9
2	5	2	7	2
9	1	9	4	9

Espejo

Figura 37.6: Distintos tipos de relleno del borde

37.4.1. 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 volúmen 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. 37.7).

37.5. 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

Figura 37.7: Desplazamiento 2x2 del filtro. El punto es el centro de la zona en la que se aplica el filtro en cada momento.

y a la vez limitaciones es que mantiene la localización espacial de las características. Aunque es necesario cierta información espacial como, por ejemplo, saber que hay presentes 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* que seleccionan el máximo de los valores u obtienen su media respectivamente (Fig. 37.8). El tamaño más típico es 2x2.

Figura 37.8: Resultado de emplear dos métodos de agrupación diferentes para reducir la dimentsión de los datos

37.6. 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. 37.9). El extractor 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 37.9: 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 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 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.

37.7. 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. 37.10).

Figura 37.10: 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. 37.11).

- 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 todos los pesos de la matriz y multiplicado por un valor α que es otro hiperparámetro que debe ser seleccionado manualmente:

$$\alpha||\boldsymbol{W}||_1 = \alpha \sum_i \sum_j |w_{ij}|. \tag{37.4}$$

Figura 37.11: Selección del modelo antes del sobreajuste

- Regularización L2 o *weight decay*. Parecida a la regularización L2 pero con una expresión algo diferente:

$$\frac{\alpha}{2}||\boldsymbol{W}||_2^2 = \frac{\alpha}{2}\sum_i \sum_j w_{ij}^2. \tag{37.5}$$

37.8. 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 *data augmentation*.

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.

Para agregar diferentes tipos de *data augmentation* en **R**, se puede hacer mediante la inclusión de capas de preprocesado 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:

Figura 37.12: Ejemplos de técnicas de generación de datos artificiales

```
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 preprocesado 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

37.9. Ejemplo 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 36, 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 capítulo 36, pero adaptando la red al tipo de dato utilizado.

37.9.1. 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)
  })
```

37.9.2. Preprocesamiento

Una vez cargados los datos y comprobado su contenido, al igual que se explicó en el capítulo 36, 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 capítulo 36, 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
```

37.9.3. 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 = 64)
#> Model: "sequential_2"
#> ________________________________________________________________
#>  Layer (type)                Output Shape              Param #
```

```
#> ================================================================
#>  conv2d_2 (Conv2D)           (None, 30, 30, 32)        896
#>  max_pooling2d_1 (MaxPoolin  (None, 15, 15, 32)        0
#>  g2D)
#>  conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496
#>  max_pooling2d (MaxPooling2  (None, 6, 6, 64)          0
#>  D)
#>  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 *4x4x64* 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 = 64)
#> Model: "sequential_2"
#> ----------------------------------------------------------------
#>  Layer (type)                Output Shape              Param #
#> ================================================================
#>  conv2d_2 (Conv2D)           (None, 30, 30, 32)        896
#>  max_pooling2d_1 (MaxPoolin  (None, 15, 15, 32)        0
#>  g2D)
#>  conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496
#>  max_pooling2d (MaxPooling2  (None, 6, 6, 64)          0
#>  D)
#>  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
#> Non-trainable params: 0
#> _________________________________________________________________
```

NOTA
Un detalle a tener en cuenta con respecto al ejemplo del capítulo 36 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
)
```

37.9.4. 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 capítulo 36 (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)
```

Figura 37.13: Evolución durante el entrenamiento de la precisión y la pérdida de los conjuntos de entrenamiento y validación

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

37.9.5. 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
```

37.9.6. Otros ejemplos de interés

- Fine tuning CNN (MobileNet, ejemplo perros y gatos) https://tensorflow.rstudio.com/guide/keras/examples/fine_tuning/
- Regresión básica con redes neuronales (conjunto de datos Boston Housing) https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_regression/
- Transfer learning en **R** https://tensorflow.rstudio.com/tutorials/advanced/images/transfer-learning-hub/

https://tensorflow.rstudio.com/

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 VIII

Ciencia de datos de texto y redes

Capítulo 38

Minería de textos

Víctor Casero, Ángela Celis y María Lozano Zahonero[1]

38.1. Introducción

En la actualidad, entre el 80 % y el 90 % de los datos que se generan diariamente son datos no estructurados (vistos en el capítulo 7). 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 en 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.

[1]Universidad de Castilla-La Mancha, Universidad de Castilla-La Mancha y Università di Roma Tor Vergata

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 el segundo apartado, se presentan los conceptos y fases fundamentales de la minería de textos, mencionando algunos paquetes de R que permiten realizar análisis textuales de distintos tipos. El tercer apartado está dedicado al análisis de sentimientos, que constituye uno de los campos de la minería de textos de mayor desarrollo en la actualidad. Cierra el capítulo un caso práctico, en el que se aplica y se amplía lo estudiado anteriormente. Unas útiles referencias sobre el tema son Fradejas Rueda (2019) y Jockers (2014).

38.2. 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 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 requieren metadatos con detalles adicionales.

38.2.1. 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 **stopwords** o **palabras vacías**. 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 el apartado 38.2.3. 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 permitir su eliminación. En el caso aplicado, 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.

38.2.2. 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 **token**, 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 *token* 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 caso aplicado, un primer análisis del significado consiste en eliminar las palabras vacías y obtener las frecuencias[2] de las palabras con valor informativo para responder a la pregunta '¿Qué se dice?' (Silge and Robinson, 2017).

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-trigramas (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 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 caso aplicado, se verá en la práctica la segmentación en *n-gramas* y cómo la visualización de redes contribuye a complementar el análisis.

***Stemming* y lematización**
La tokenización se puede refinar mediante el *stemming*, o reducción de las palabras 'flexionadas'

[2] Frecuencias relativas si se comparan distintos textos. También será útil para ello obtener la tasa de riqueza léxica (TIR).

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.

38.2.3. Campos de aplicación de la minería de textos

La minería de textos tiene varios campos de aplicación. Entre ellos destacan tres: el **análisis de sentimientos**, el **modelado de temas** o **tópicos** y el **análisis estilométrico** o **estilometría**. El *análisis de sentimientos* se tratará con detalle en la sección 38.3 y en el caso de aplicación (sección 38.4.4), mientras que el modelado de temas o tópicos, se ilustrará en el capítulo 58. Dadas las limitaciones del manual sólo se presentan brevemente el modelado de temas y la estilometría.

El **modelado de temas** (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*).

La **estilometría** 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 su sigla 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.

38.2.4. 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 integrado, 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 Gutenberg[3]; muy útil si se quieren analizar textos literarios.

38.3. 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*.

[3]Proyecto 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.

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 caso aplicado.

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 disponibles 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 caso aplicado 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.

38.4. Caso de aplicación

38.4.1. Declaración institucional del estado de alarma 2020

La 'Declaración institucional del presidente del gobierno sobre 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. Ésta se puede encontrar en el paquete `CDR` que acompaña este libro. Se le van a aplicar las operaciones y técnicas mencionadas en la sección 38.2.

```
library("CDR")
data("declaracion")
# load("data/declaracion-estado-alarma.Rdata")
```

38.4.2. Segmentación en palabras y oraciones

Las primeras tareas del análisis son la preparación, limpieza y segmentación o tokenización de los textos. 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 pueden 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.

```
tabla <- table(palabras[[1]])
(tabla <- tabla |>
  tibble(palabra = names(tabla), recuento = as.numeric(tabla)) |>
  arrange(desc(recuento)))
#> # A tibble: 390 x 3
#>    tabla       palabra recuento
#>    <table[1d]> <chr>      <dbl>
#>  1 43          de            43
#>  2 41          y             41
#>  3 35          la            35
#>  4 31          a             31
#>  5 26          los           26
#>  6 22          en            22
#>  7 20          que           20
#>  8 17          el            17
#>  9 14          al            14
#> 10 14          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 el siguiente apartado, 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
#> [1] "Buenas tardes."
```

También podría medirse la longitud de cada oración, en número de palabras, típicamente para comparaciones con otros textos. Para ello se debe separar cada oración en palabras y con la función `sapply` se puede obtener la longitud de cada oración, que puede verse en la Figura 38.1.

```
palabras_oracion <- tokenize_words(oraciones[[1]])
longitud_o <- sapply(palabras_oracion, length)
head(longitud_o)
#> [1]  2  2 39 33 33 32
```

38.4.3. Análisis exploratorio

Eliminación de palabras vacías
Se va a hacer uso del 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):

Figura 38.1: Número de palabras en cada oración de la Declaración

Tabla 38.1: Palabras más frecuentes (sin palabras vacías)

tabla	palabra	recuento
9	virus	9
7	recursos	7
5	social	5
4	alarma	4
4	conjunto	4
4	emergencia	4
4	españa	4
4	semanas	4
4	va	4
3	cada	3

```
tabla <- tabla |> anti_join(tabla_stopwords)
knitr::kable(tabla[1:10, ],
  caption = "Palabras más frecuentes (sin palabras vacías)"
)
```

El resultado, Tabla 38.1, 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 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 (2019), o pueden

confeccionarse listas *ad hoc*.

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 contener 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 38.2: Nube de palabras más frecuentes de la Declaración

El resultado se muestra en la Figura 38.2. Como se puede observar, el tamaño de letra de la palabra, y en este caso también el color, están relacionados con su frecuencia.

38.4.4. Análisis de sentimientos y detección de emociones

Lexicón `bing`
El lexicón `bing`, como se ha visto en la sección 38.3, 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 descri-

Tabla 38.2: 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

to arriba de preparación, limpieza, segmentación en palabras, eliminación de palabras vacías (obviamente, en idioma inglés).

```
data("EN_declaracion")
# declaracion_EN <-
↪  paste(read_lines("data/EN-declaracion-estado-alarma.txt"),collapse = "\n")

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 38.2.

Lexicón NRC

Para poder observar las similitudes y diferencias en el análisis según el lexicón elegido, se aplica también NRC a la Declaración (véase la Tabla 38.2).

Con el léxico NRC pueden detectarse emociones. La misma palabra puede tener asociada distintas emociones/sentimientos. En la Figura 38.3 se puede observar la dispar frecuencia de palabras de cada tipo:

```
emo <- get_sentiments("nrc")
emo |> ggplot(aes(sentiment)) +
  geom_bar(aes(fill = sentiment), show.legend = FALSE)
```

Figura 38.3: 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 en el que 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
#> 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 gráfico de barras (Figura 38.4) bien con nubes de palabras (Figura 38.5).

```
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 38.4: 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 emociones[4]:

```
library(syuzhet)
palabras_EN2 <- get_tokens(EN_declaracion)
emo_tab2 <- get_nrc_sentiment(palabras_EN2, lang = "english")
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")
```

[4]Se deja al lector el análisis de la Declaración con más emociones, en castellano, etc.

```
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 38.5: Nube de palabras de tres emociones NRC seleccionadas

38.4.5. *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,
  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:

```
bigramas_limpios <- bigramas |>
  tidyr::separate(bigram, c("word1", "word2"), sep = " ") |>
  dplyr::filter(!word1 %in% tabla_stopwords$palabra) |>
  dplyr::filter(!word2 %in% tabla_stopwords$palabra) |>
  tidyr::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
```

Significado y contexto
Como se ha visto en la sección 38.2.2, 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 |>
  tidyr::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.

38.4.6. Análisis de redes

Se proporcionan las instrucciones para realizar un análisis básico de redes, utilizando los paquetes `igraph` y `ggraph`. Dada la corta extensión de la Declaración no es posible obtener conclusiones. En la figura 38.6 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_l <- bigramas_limpios |>
  tidyr::separate(bigram, c("first", "second"), sep = " ") |>
  dplyr::filter(n > 1) |>
  graph_from_data_frame()
g1 <- ggraph(graf_bigramas_l, layout = "fr") +
  geom_edge_link() +
  geom_node_point(size = 0) +
  geom_node_text(aes(label = name))
graf_bigramas <- bigramas |>
  tidyr::separate(bigram, c("first", "second"), sep = " ") |>
```

```
  dplyr::filter(n > 2) |>
  graph_from_data_frame()
g2 <- ggraph(graf_bigramas, layout = "fr") +
  geom_edge_link() +
  geom_node_point(size = 0) +
  geom_node_text(aes(label = name))
library(patchwork)
g1 + g2
```

Figura 38.6: Redes de bigramas sin palabras vacías y con ellas

Resumen

En este capítulo se ha introducido al lector en la minería de textos, en particular:

- Se han presentado los conceptos y tareas fundamentales de este ámbito, así como sus campos de aplicación principales. Se ha puesto 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 ha mostrado el uso de `R` para el análisis de textos y de sentimientos.
- Se ha presentado un caso aplicado para ilustrar las técnicas de minería de textos.
- Se han mencionado otros análisis plausibles de minería de textos como la estilometría o el modelado de temas (véase el capítulo 58).

Capítulo 39

Análisis de grafos y redes sociales

José Javier Galán

39.1. Introducción

El origen de la teoría de grafos se debe al problema de los Siete puentes de Königsberg (Paul Euler, 1736), 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.39.1). Euler mediante el uso de grafos demostró que no era posible.

Figura 39.1: Siete puentes de Königsberg, Euler.

39.2. Teoría de grafos

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 puede representar mediante grafos una **red social** (ver Fig.39.2).

```
# Cargar el paquete de igraph
# if(!require('igraph')) install.packages('igraph')
library(igraph)

datos_facebook <- read.csv("data/DatosFacebook.csv", header = FALSE, sep = "
↪  ", col.names = c("Origen", "Destino"))
grafo_facebook <- graph.data.frame(datos_facebook, directed = T)
plot(grafo_facebook, vertex.label = NA)
```

Figura 39.2: Estructura de una red social representada como grafo.

Una **red social** es una estructura de red invisible 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.

39.3. Elementos de un grafo

El análisis de redes sociales mediante le teoría de grafos requiere conocer previamente una serie de conceptos básicos, (Perez Sola, 2021).

- Las **aristas** son la relaciones que unen los nodos. Son **dirigidas** (ver Fig.39.3) si tienen un sentido definido y **no dirigidas** (ver Fig.39.4) en caso contrario.

```
datos_grafo <- read.csv("data/DatosGrafos.csv", sep = ";")
grafo <- graph.data.frame(datos_grafo, directed = T)
plot(grafo, edge.label = paste(E(grafo)$weight))
```

Figura 39.3: Grafo dirigido.

```
grafo <- graph.data.frame(datos_grafo, directed = F)
plot(grafo, edge.label = paste(E(grafo)$weight))
```

Figura 39.4: Grafo no dirigido.

En una red social como **LinkedIn** las aristas representan la relación que une las personas. Las personas forman parte de un grupo con intereses comunes, formando un grado no dirigido.

Pero también pueden seguir a alguien sin necesariamente ser seguido, en ese caso podemos representar un grafo dirigido.

- Los **vértices**, vertex , representan nodos que serán unidos mediante aristas. En una red social cada vértice representa una de las personas de dicha red, unidas en ocasiones por intereses comunes a otras.
- Un **grafo** es un conjunto de vértices y de aristas que podemos 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.
- Una **matriz de adyacencia** representa una matriz cuadrada de $n \times n$, siendo n el número de vértices del grafo. Se inicia con valor 0 donde a cada valor a_{ij} se suma 1 por cada arista que relaciona los vértices i y j. Si no existe relación el elemento a_{ij} vale 0.

```
matriz_adyacencia <- get.adjacency(grafo, sparse = FALSE)
matriz_adyacencia
#>   1 2 3 4 5
#> 1 0 1 1 1 1
#> 2 1 0 1 2 0
#> 3 1 1 0 2 1
#> 4 1 2 2 0 0
#> 5 1 0 1 0 0
```

- El **grado** de un nodo son el numero de aristas que tiene dicho nodo x, y se representa mediante grado(x), g(x) o gr(x), siendo un vértice de grado 0 un vértice aislado. En un grafo G existirá un grado máximo $\Delta(G)$ y un grado mínimo $\delta(G)$, mientras que el grado del grafo, g(G), es la suma de todos los grados de sus vértices. En una red social representa el número de relaciones que existen, en una red social como Facebook significaría conocer cuántos amigos tienes.

```
degree(grafo)
#> 1 2 3 4 5
#> 4 4 5 5 2
```

- Un **camino** une dos vértices constituyendo un conjunto de aristas no recursivas. Entre dos vértices puede no existir un camino, puede haber varios y se puede incluir el mismo vértice en el camino mas de una vez. Evidentemente siempre habrá un **camino mas corto** , aquel que menos aristas ha recorrido. Si entre todos los pares de vértices existe un camino estamos hablando de un grafo conexo.

```
# Camino más corto entre el nodo 2 y el 5
caminos <- get.shortest.paths(grafo, from = "2", to = "5")
V(grafo)[caminos$vpath[[1]]]
#> + 4/5 vertices, named, from d9b4a59:
#> [1] 2 4 3 5
```

39.4. El paquete igraph

Existen diversos paquetes, pero el más utilizado y popularizado por sencillez y eficacia es `igraph` (igraph.org, 2022). Se trata de un paquete que permite crear y manipular grafos para poder analizar redes en **R** de forma muy sencilla (ver Fig.39.5).

```
nodes <- cbind("nodos" = c("A", "B", "C", "D", "E"))
edges <- cbind("from" = c("A", "C", "B", "A", "A", "A"), "to" = c("B", "D",
↪  "C", "C", "D", "E"))
red <- graph_from_data_frame(edges, directed = F, vertices = nodes)

plot(red)
```

Figura 39.5: Ejemplo de grafo con Igraph.

Para crear un grafo a partir de un **dataframe** se han usado los argumentos:

`graph_from_data_frame(edges, directed = TRUE, vertices = NULL)` donde:

- **edges** Es un data frame donde las dos primeras columnas representan una lista de aristas.
- **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 vertices o NULL.

```
library(igraph)
nodes <- cbind("actores" = c("Jim Carrey", "Arnold Swarzenneger", "George
↪  Cloney", "Cameron Diaz"), "descripcion" = c("actor", "actor", "actor",
↪  "actriz"))
edges <- cbind(
```

```
  "from" = c("Jim Carrey", "Jim Carrey", "George Cloney", "Jim Carrey"),
  "to" = c("Arnold Swarzenneger", "George Cloney", "Arnold Swarzenneger",
  ↪  "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)
```

Figura 39.6: Grafo representativo de la relacion de actores respecto a peliculas

En la Fig. 39.6 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 participo con uno de ellos.

39.5. Análisis de influencia en un grafo aplicado a redes sociales

Existen paquetes para obtener informacion de distintas redes sociales, por ejemplo, se puede utilizar el paquete `Rfacebook` para conectarse a **Facebook** y obtener información de lo contactos existentes, para ello sera necesario activar la API desde **https://developers.facebook.com**. La información necesaria podemos encontrarla en su página web **https://developers.facebook.com/docs**. Para ilustrar un ejemplo didactico, sin ncesidad de que el lector necesite conocimientos de desarrollo para descargar datos, se ha generado un fichero excel que simula la relacion entre amigos de una red social como podria ser perfertamente Facebook.

- Se incorporan los datos y se muestra su cabecera. Los datos han sido recogidos en un fichero csv, constando de dos columnas separadas por espacio. La primera columna contiene el

id de la primera persona, la cual establece relación con la persona indicada también por su id en la segunda columna.

```
datos_faceboook <- read.csv("data/DatosFacebook.csv", header = FALSE, sep = "
↪  ", col.names = c("Origen", "Destino"))
head(datos_faceboook)
#>   Origen Destino
#> 1   3434    3409
#> 2   3493    3361
#> 3   3329    3324
#> 4   3496    3384
#> 5   3370    3415
#> 6   3383    3490
```

Un grafo no tiene una centralidad real porque no tiene coordenadas, pero tenemos distintas medidas de centralidad que en un grafo social nos ayudara a identificar el poder social de cada individuo, es decir, su influencia.

Se puede decir que la **centralidad de un grafo** coincide con el grado de vértices a analizar, si un vértice representa una persona de una red social de amigos su centralidad representa el número de amigos que tiene.

En los grafos dirigidos cuantas más aristas dirigidas a un nodo existan significará que más personas intentar interactuar con una persona concreta y esta más prestigio tendrá. 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.

- Se convierten los datos a formato igraph y mostramos el grafo (ver Fig.39.7). Se compone el data frame de tipo igraph necesario indicando que es un grafo dirigido, finalmente mediante 'plot' se muestra el grado al mismo tiempo que se establecen sus propiedades.

```
grafo_faceboook <- graph.data.frame(datos_faceboook, directed = T)

plot.igraph(grafo_faceboook,
  layout = layout.fruchterman.reingold,
  vertex.label = NA,
  vertex.label.cex = 1,
  vertex.size = 3, edge.curved = TRUE,
  edge.arrow.size = 0
)
```

- Relaciones de la red social. Se puede observar que el numero de aristas que contiene cada vertice son el numero de personas con las que se relaciona cada individuo, por lo tanto son las relaciones que contiene cada persona.

Figura 39.7: Aplicación de Igraph a Redes Sociales.

```
table(degree(grafo_faceboook))
#>
#>  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
```

- Se personalizan los datos (ver Fig.39.8). Son muchos nodos y para focalizar este caso de estudio se centra el codigo en los nodos que sean de grado igual o superior a 26 y se les asigna nombre.

```
bad_network <- V(grafo_faceboook)[degree(grafo_faceboook) <= 26]
grafo_faceboook <- delete.vertices(grafo_faceboook, bad_network)
V(grafo_faceboook)$name <- c("Gema", "Patricia", "Ramon", "Jose", "Maria",
↪  "Angeles", "Gabriel", "Javier", "Victor", "Leonor", "Ana", "Isabel",
↪  "Cristobal", "Rosa", "Aurola")
plot(grafo_faceboook)
```

Se pueden observar las relaciones entre las distintas personas que lo componen, como siguen a otras y son seguidas. Tambien se pueden observar los casos extremos como el de Gema a quien nadie sigue, o el de Leonor que no sigue a nadie pero varios podrian llegar hasta ella.

Detección de comunidades

En el análisis de una red social es importante detectar las distintas comunidades que la componen.

- Se puede observar dos comunidades detectadas con **Walktrap** en la (ver Fig.39.9)

Figura 39.8: Aplicación de Igraph a Redes Sociales, grafo dendrograma

```
Walktrap_grafo <- walktrap.community(grafo_faceboook, steps = 5, modularity =
↪  TRUE)
plot(Walktrap_grafo, grafo_faceboook, edge.arrow.size = 0.25, vertex.label =
↪  (grafo_faceboook)$name)
```

Figura 39.9: Walktrap Community on Full dataset

```
dend_g_network <- as.dendrogram(Walktrap_grafo)
plot(dend_g_network)
```

Walktrap detecta las comunidades basandose en el concepto de que las caminatas aleatorias

Figura 39.10: Grafo dendrograma de Walktrap

cortas permanecen en la misma comunidad. Debe indicarse èl largo de la caminata aleatoria, se recomienda usar 5 caminatas. Se puede ver su dendrograma en la Fig.39.10. El dendograma es un tipo de diagrama en forma de arbol, en el cual se subdivide los datos en subcategorias sucesivamente hasta llegar a las vistas deseadas.

- La técnica de **Centralidad de intermediación (betweenness)** se basa en el número de caminos mínimos en los que un nodo esta involucrado. Por lo tanto, en un red social una persona tendrá mayor influencia cuanto mayor betweenness tenga porque comunicara mucha información a través de los nodos de la red. Si puede llegar a un grupo grande, aunque sea a través de un nodo puede alcanzar un gran nivel de viralización (ver Fig.39.11 y Fig.39.12).

```
edge_g_network <- edge.betweenness.community(grafo_faceboook)
plot(as.dendrogram(edge_g_network), main = "Grafo dendrograma de Betweenness")
```

```
plot(edge_g_network, grafo_faceboook, edge.arrow.size = 0.25, main = "Edge
↪  Betweenness Community")
```

En este ejemplo vemos como se han detectado mediante este algoritmo siete comunidades.

- **Centralidad de vector propio (eigenvector)** se basa en la centralidad de los nodos con los que se relaciona, la centralidad de un nodo es proporcional a la suma de las centralidades de sus nodos vecinos.

Figura 39.11: Grafo dendrograma de Betweenness

Figura 39.12: Grafo dendrograma de Betweenness

```
cl_g_network <- leading.eigenvector.community(grafo_faceboook)
plot(cl_g_network, grafo_faceboook, edge.arrow.size = 0.25, main = "Leading
↪  Eigenvector Community")
```

Figura 39.13: Aplicación de Igraph a Redes Sociales

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.

En este ejemplo vemos como se han detectado mediante este algoritmo tres comunidades. Segun este metodo, no es tan importante que tengas muchos amigos, lo importante es que tus amigos sean muy influyentes (ver Fig.39.13).

39.6. Otras utilidades de grafos

En la Fig 39.14 se muestran los tipos de accidente ocurridos en Madrid durante el año 2020 que tiene como relación la zona en la que sucedió, en este caso en "Barajas". Ademas se añade a las aristas el numero de accidentes sucedido por cada tipo.

```
library(igraph)
library(CDR)

datos_accidentalidad_2020 <- accidentes2020_data
conjunto <- datos_accidentalidad_2020[, c("distrito", "tipo_accidente")]

distrito <- datos_accidentalidad_2020[, c("distrito")]
tipo_accidente <- datos_accidentalidad_2020[, c("tipo_accidente")]
```

```
grupos <- group_by(conjunto, distrito, tipo_accidente)
contados <- dplyr::summarise(grupos,
  num = n()
)

zona_Barajas <- dplyr::filter(contados, distrito == "BARAJAS")
g3 <- graph_from_data_frame(zona_Barajas)
plot.igraph(g3, edge.label = zona_Barajas$num)
```

Figura 39.14: Accidentes en Barajas en el año 2020

Ahora mostramos los principales puntos conflictivos de Barajas unidos para ver si existen caminos que pasaern por otros accidentes y evitarlos, Fig 39.15.

```
library(proj4)
library(igraph)
library(sp)
library(leaflet)

library(CDR)
datos <- accidentes2020_data
datos <- datos[, c("localizacion", "distrito", "coordenada_x_utm",
↪  "coordenada_y_utm")]
datos <- dplyr::filter(datos, distrito == "BARAJAS")
datos <- datos[, c("localizacion", "coordenada_x_utm", "coordenada_y_utm")]
datos <- distinct(datos)
```

```
meta <- data.frame(name = datos[["localizacion"]], lon =
↪  datos[["coordenada_x_utm"]], lat = datos[["coordenada_y_utm"]])
proj4string <- "+proj=utm +zone=30 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
↪  "


xy <- data.frame(meta$lon, meta$lat) # data frame con utm
pj <- project(xy, proj4string, inverse = TRUE)
meta <- data.frame(name = datos[["localizacion"]], lat = pj$y, lon = pj$x)
df <- data.frame(from = datos[["localizacion"]], to = datos[["localizacion"]])
meta <- head(distinct(meta))
df <- head(distinct(df))


mat <- matrix(ncol = 0, nrow = 0)
df2 <- data.frame(mat)
for (f in df[, 1])
{
  for (t in df[, 2])
  {
    fila_nueva <- c(f, t)
    df2 <- rbind(df2, fila_nueva)
  }
}
df <- df2

g <- graph.data.frame(df, directed = FALSE, vertices = meta)
gg <- get.data.frame(g, "both")
vert <- gg$vertices
coordinates(vert) <- ~ lon + lat

edges <- gg$edges

edges <- lapply(1:nrow(edges), function(i) {
  as(
    rbind(
      vert[vert$name == edges[i, "from"], ],
      vert[vert$name == edges[i, "to"], ]
    ),
    "SpatialLines"
  )
})


for (i in seq_along(edges)) {
```

```
  edges[[i]] <- spChFIDs(edges[[i]], as.character(i))
}

edges <- do.call(rbind, edges)

leaflet(vert) %>%
  addTiles() %>%
  addMarkers(data = vert) %>%
  addPolylines(data = edges)
```

Figura 39.15: Grafo sobre mapa interactivo

RESUMEN
El uso de los grafos para el estudio de las redes sociales resultad de gran utilidad y aunque existen distintos paquetes uno de los mas potentes sigue siendo `igraph`. Se ha aprendido que tres componentes son necesarios: obtener datos de una red social, conocer las propiedades de los grafos y aplicarlos mediante el paquete `igraph`. Gracias a los grafos podemos representar las relaciones que unen a los individuos en una red social, asi como las comunidades que forman e incluso la popularidad de cada uno, sus seguidores y a quien siguen.

Parte IX

Ciencia de datos espaciales

Capítulo 40

Trabajando con datos espaciales

Gema Fernández-Avilés[1]

Los datos espaciales, también conocidos como datos geográficos, 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) y la ubicación de las bombas de agua en el área (ver Fig. 40.1). 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 área que se encarga de estudiar y analizar los datos espaciales es la **estadística espacial**.

El análisis espacial de Snow es considerado un caso típico 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.

40.1. 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. 40.2, representa la localización de los accidentes de tráfico registrados en la ciudad de Madrid durante el año 2020. Sin embargo, la representación no aporta información útil para su análisis.

[1]Quisiera agradecer a Diego Hernangómez toda la ayuda prestada en la elaboración de este capítulo.

Figura 40.1: Mapa de cólrela en Londres según Snow. Fuente: Wikipedia

```
library(CDR)
library(ggplot2)
ggplot(
  data = accidentes2020_data,
  aes(x = coordenada_x_utm, y = coordenada_y_utm)
) +
  geom_point(col = "blue")
```

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 proyectadas las coordenadas 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
)

library(mapSpain)
madrid <- esp_get_munic(munic = "^Madrid$") |>
  st_transform(25830)
```

Figura 40.2: Accidentes de Tráfico en Madrid (2020)

```
tile <- esp_getTiles(madrid, "IDErioja", zoommin = 2) # Imagen

ggplot() +
  tidyterra::geom_spatraster_rgb(data = tile) +
  geom_sf(
    data = accidentes2020_sf,
    col = "blue", size = 0.3, alpha = 0.3
  )
```

Figura 40.3: Accidentes de Tráfico en Madrid proyectados y con mapa de carreteras (2020)

La Fig.40.3 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 las autopistas de salida de la ciudad.

40.1.1. Sistema de Referencia de Coordenadas

Los CRS que 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.

En la Fig. 40.4 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)
library(sf)
paises <- gisco_get_countries()
puertos <- gisco_get_ports()
paises_robin <- st_transform(paises, st_crs("ESRI:54030")) #+proj=robin

plot(st_geometry(paises_robin), main = " ")
plot(st_geometry(puertos), add = TRUE, col = "2", pch = 20, lwd = 2.5)
```

Figura 40.4: Localización de los puertos en el mapa mundi (distinto CRS puertos y mapa)

Los dos grandes tipos de CRS son:

(I) **Geográficos**: aquellos en los que los parámetros empleados para localizar una posición espacial son la latitud (Norte-Sur [-90º,90º]) y la longitud (Este-Oeste [-180º,180º]). En este caso las distancias entre dos puntos representan **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 mapa mundi (Robinson), la Fig. 40.5 muestra adecuadamente el mapa de la Fig. 40.4.

```
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 40.5: 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 `crsuggest` (Walker, 2022) facilita la labor, sugiriendo el CRS más adecuado para cada zona.

40.1.2. Formatos de datos espaciales

En el ámbito del análisis espacial se pueden clasificar el formato de datos espaciales en función del modelo de datos . Se pueden distinguir dos tipos de modelos de datos (Lovelace et al., 2019): vectores y ráster[2].

[2]Un análisis detallado puede verse en Hernangómez and Fernández-Avilés (2022)

40.1.2.1. 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. **??**).

```
library(ggplot2)
library(sf)
library(CDR)

ggplot() +
  geom_sf(
    data = hosp_toledo,
    aes(fill = "Hospitales y Centros Sanitarios"),
    color = "blue"
  ) +
  labs(title = " ", fill = " ") +
  theme_minimal() +
  theme(legend.position = "right")
```

```
{plot-puntos, echo=FALSE, fig.cap="Hospitales y Centros de Salud en Toledo"}
knitr::include_graphics(img/centros-toledo.PNG")
```

Los puntos también pueden estar conectados entre sí, de manera que formen geometrías más complejas, como **líneas** y **polígonos**.

En la Fig. 40.6 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).

```
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()
```

Las extensiones más habituales de los archivos que contienen datos de vectores se muestran a continuación:

Tabla 40.1: Ficheros con datos vector

Tipo	Extensión
Shapefile	`.shp`, `.shx`, `.dbf`
GeoPackage vector	`.gpkg`

Tipo	Extensión
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 quizá el formato más empleado. Sin embargo, tiene una serie de limitaciones: es un formato multiarchivo y el CRS es opcional.

40.1.2.2. Datos ráster

Los datos raster son datos proporcionados en una rejilla rectangular de píxeles regulares o no (denominada **matriz**). El caso más cotidiano de un ráster es una fotografía, donde la imagen se representa como una serie de celdas, determinadas por la resolución de la imagen (número total de píxeles) y 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. 40.7:

```
library(terra)
library(CDR)
library(sf)
elev <- rast(system.file("external/Toledo_DEM.asc", package = "CDR"))
plot(elev, main = " ")
pols <- as.polygons(elev)
plot(pols, add = TRUE, border = "grey90")
plot(st_geometry(Tol_prov), add = TRUE)
```

En la Fig. 40.7, 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 40.2: Ficheros con datos raster

Tipo	Extensión
ASCII Grid	`.asc`
GeoTIFF	`.tif`, `.tiff`
Enhanced Compression Wavelet	`.ecw`

Figura 40.6: Datos vector: Puntos, líneas y polígonos

Figura 40.7: Datos ráster. Elevación de la provincia de Toledo

40.2. Mi primer mapa

Definidas las principales características 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 España[3].

Ambos conjuntos deben tener, al menos, un campo en común, `codigo_ine` en este caso, para su unión.

```
library(sf)
library(CDR)
munis_renta <- municipios |>
  # unión de datasets
  dplyr::left_join(renta_municipio_data) |>
  # selección de variables
  dplyr::select(name, cpro, cmun, `2019`)
```

```
library(ggplot2)
ggplot(munis_renta) +
  geom_sf(aes(fill = `2019`), color = NA) +
  theme_minimal() +
  scale_fill_continuous(
    labels = scales::label_number(
      big.mark = ".", decimal.mark = ",", suffix = " €"
    )
  ) +
  theme_minimal()
```

La Fig. 40.8 presenta una serie de elementos gráficos, característicos de los objetos espaciales, tal y como puede verse en la estructura del 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
#> 1     Abla   04  001 10192 MULTIPOLYGON (((-2.775594 3...
#> 2 Abrucena   04  002 10021 MULTIPOLYGON (((-2.787566 3...
#> 3     Adra   04  003  8192 MULTIPOLYGON (((-3.051988 3...
```

[3]Un análisis detallado puede verse en (Hernangómez and Fernández-Avilés, 2022)

Figura 40.8: Distribución de la renta neta media por persona (€) en 2019

40.3. ¿Cómo (no) mentir con la visualización?

Si se realiza un mapa de coropletas como el de la Fig. 40.8 puede que la información quede 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 Gamma) y será necesario dividir los datos en clases con el paquete **classInt** (Bivand, 2020). Se utiliza el método de Fisher-Jenks, desarrollado específicamente para la clasificación de datos espaciales y su visualización en mapas. Además, se eliminan los municipios sin datos y se elige una escala de color adecuada. El mapa de la Fig. 40.9 proporciona ahora una visualización adecuada de la variable renta.

```
library(sf)
# crea etiquetas formateadas
label_fun <- function(x) {
  l <- length(x)
  eur <- paste0(prettyNum(round(x, 0),
    decimal.mark = ",", big.mark = "."
  ), " €")
  labels <- paste(eur[-l], "-", eur[-1])
  labels[1] <- paste("<", eur[1])
```

```
  labels[l - 1] <- paste(">", eur[l - 1])
  return(labels)
}

library(classInt)
munis_renta_clean <- munis_renta |>
  dplyr::filter(!is.na(`2019`))

# crea Fisher-Jenks clases
fisher <- classIntervals(munis_renta_clean$`2019`,
  style = "fisher", n = 10
)

breaks_f <- fisher$brks
labels_f <- label_fun(breaks_f)

munis_renta_clean$`Fisher-Jenks` <- cut(munis_renta_clean$`2019`,
  breaks = breaks_f, labels = labels_f, include.lowest = TRUE
)

# dibuja renta municipal
ggplot(munis_renta_clean) +
  geom_sf(aes(fill = `Fisher-Jenks`), color = NA) +
  scale_fill_manual(values = hcl.colors(length(labels_f),
    "Inferno",
    rev = TRUE
  )) +
  theme_minimal()
```

40.4. 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. 40.11 representa la temperatura mínima registrada en España del 6 al 10 de Enero de 2021, `tempmin_data`, durante la borrasca Filomena.

```
library(sf)
tmin_sf <- st_as_sf(tempmin_data,
  coords = c("longitud", "latitud"),
  crs = 4326 # coordenadas geográficas longitud/latitud
)
library(mapSpain)
esp <- esp_get_ccaa() |> # sf objeto, contorno de España
  filter(ine.ccaa.name != "Canarias") # excluye Canarias
```

Figura 40.9: Renta neta per cápita (€) por tramos según Fisher-Jenks

¿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
```

La Fig. 40.10 representa las estaciones de monitoreo.

```
library(ggplot2)
ggplot(esp2) +
  geom_sf() +
  geom_sf(data = tmin_sf) +
  theme_light()
```

NOTA Los paquetes **tmap** (Tennekes, 2018) y **mapsf** (Giraud, 2022) son referentes para mapas temáticos.

```
# definición de intervalos
cortes <- c(-Inf, seq(-20, 20, 2.5), Inf)
colores <- hcl.colors(15, "PuOr", rev = TRUE)
```

Figura 40.10: 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. mím") +
  scale_color_gradientn(
    colours = colores,
    breaks = cortes,
    labels = function(x) {
      paste0(x, "º")
    },
    guide = "legend"
  )
```

40.5. Mapas interactivos

El desarrollo de la informática ha propiciado también el desarrollo de la geocomputación y está permite, entre otras cosas, la representación de mapas interactivo.

```
library(leaflet)
library(isdas)
```

Figura 40.11: Temperatura mínima en España (6-10 enero 2021)

```
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
```

A modo de ejemplo, el mapa de la Fig. 40.12 representa el mapa Fig. 40.1 de forma interactiva con la librería `leaflef`. Éstos mapas dinámicos, ampliables y desplazables son más informativos que los mapas estáticos.

40.6. Estadística para datos espaciales

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

Figura 40.12: Mapa interactivo de las muertes por cólera en Londres según Snow

Gotway, 2005, p. 6). Cressie (1993) proporciona una clasificación de datos espaciales basada en la naturaleza del dominio espacial en estudio y distingue: datos geoestadísticos, datos de patrones de puntos y datos lattice o reticulares . La Fig. 40.13 ilustra los distintos procesos espaciales que se abordarán en los capítulos XXXX.

Figura 40.13: Clasificación de datos espaciales propuesta por Cressie (1993)

RESUMEN
Los datos espaciales son aquellos que contienen información de una zona geográfica de la tierra. Éstos 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 41

Geoestadística

Gema Fernández-Avilés y José-María Montero

41.1. Introducción

El término "geoestadística" apareció por primera vez en Matheron (1962), y en él "geo" enfatiza la referencia a las Ciencias de la Tierra, mientras que "estadística" se refiere al uso de métodos probabilísticos-inferenciales.

La geoestadística estudia de 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)[1].

- 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 hacer una representación exhaustiva del fenómeno y pero sí, partir de las observaciones disponibles, reconstruirlo.

[1] Los datos geoestadísticos son tan solo una parte de los datos espaciales: Otra parte de ellos, son los datos "laticce" 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. 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 punto.

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... 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.

41.2. 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 $z(\mathbf{s_i}), i = 1, 2, 3, ...$ 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.*). Si se toman valores regionalizados en todos los puntos del dominio, D, es decir, si se considera *v.r.*, ésta puede 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, $\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_k}$, las *v.a.* $Z(\mathbf{s_1}), Z(\mathbf{s_2}), ..., Z(\mathbf{s_k})$ 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 analísis geoestadístico.

Una *f.a.* espacial es *estrictamente estacionaria* si las familias de *v.a.* $Z(\mathbf{s}_1), Z(\mathbf{s}_2), ..., Z(\mathbf{s}_k)$, tienen la misma distribución de probabilidad conjunta que $Z(\mathbf{s}_1+\mathbf{h}), Z(\mathbf{s}_2+\mathbf{h}), ..., Z(\mathbf{s}_k+\mathbf{h}), \forall k$, $\forall \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_k$ y $\forall \mathbf{h} \in \mathbb{R}^d$ (donde $\mathbf{h}$ es un vector de traslación), siempre que $\mathbf{s}_1+\mathbf{h}, \mathbf{s}_2+\mathbf{h}, ..., \mathbf{s}_k+\mathbf{h} \in D$. Es decir, la distribución de probabilidad conjunta de $Z(\mathbf{s}_1+\mathbf{h}), Z(\mathbf{s}_2+\mathbf{h}), ..., Z(\mathbf{s}_k+\mathbf{h})$ 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 practica lo que se suele asumir es la *estacionariedad de segundo orden*, que limita la estacionariedad a los dos primeros momentos de la *f.a.*[2]

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 tal importantísima 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; es decir, 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.*[3] 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.

[2]En geoestadística lineal el interés se centra en los dos primeros momentos de la *f.a.*, por cual tan solo 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 solo depende de $\mathbf{h}$.

[3]Estas realizaciones no son independientes y se suele asumir también la hipótesis de ergodicidad.

41.3. Análisis estructural de la dependencia espacial

41.3.1. 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. 41.1 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
)
```

Pasando del terreno de las simulaciones a la realidad, la Fig. 41.2 muestra la temperatura máxima en España el 6 de agosto de 2022[4], 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
```

[4]Los 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 41.1: Dependencia espacial frente a aleatoriedad

```
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 los patrones de dependencia espacial observados mediante funciones que indiquen cuál es la estructura de la dependencia espacial, para ser utilizadas en el proceso de predicción. 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.*:

$$\gamma(\mathbf{s}_i - \mathbf{s}_j) = \frac{1}{2}V[Z(\mathbf{s}_i) - Z(\mathbf{s}_j)], \forall \mathbf{s}_i, \mathbf{s}_j \in D. \tag{41.1}$$

Figura 41.2: Temperatura máxima en España peninsular, 6 de agosto de 2022

que, en el caso habitual de *f.a.* estacionarias de segundo orden o intrínsecamente estacionarias (sin deriva), se transforma en:

$$\gamma(\mathbf{h}) = \frac{1}{2}V\left(Z(\mathbf{s}+\mathbf{h}) - Z(\mathbf{s})\right) = \frac{1}{2}E\left(\left(Z(\mathbf{s}+\mathbf{h}) - Z(\mathbf{s})\right)^2\right), \tag{41.2}$$

Nótese que:

- Si hay dependencia espacial (positiva[5], 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 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

[5] Valores cercanos similares. Si es negativa, los valores vecinos son diferentes.

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).

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*.

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, $lim_{|\mathbf{h}|\to\infty}\frac{\gamma(\mathbf{h})}{|\mathbf{h}|^2} = 0$ 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.

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. 41.3, 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. 41.3, 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())
```

Figura 41.3: Representacion bidimensional de dos f.a.: Con semivariograma parabólico en el origen (izquierda); con semivariograma lineal en el origen (derecha)

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 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. 41.4 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 41.4: Elementos del semivariograma (meseta unitaria)

41.3.2. Modelos de semivariogramas válidos

Las funciones que verifican las condiciones que debe cumplir un semivariograma (véase 41.3.1) 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:[6]

41.3.2.1. Semivariogramas con meseta

Están asociados a *f.a.* estacionarias de segundo orden. Los más utilizados son:

[6] Aquí se presentan los semivariogramas isotrópicos más utilizados en la práctica. Nos centramos en ellos porque 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 pueden representarse separadamente mediante semivariogramas isotrópicos. Para un análisis detallado, consúltese, por ejemplo, Montero et al. (2015).

- **Esférico.** Válido en $\mathbb{R}^1$, $\mathbb{R}^2$ y $\mathbb{R}^3$, y viene dado por:

$$\gamma(|\mathbf{h}|) = \begin{cases} m\left(1{,}5\frac{|\mathbf{h}|}{a} - 0{,}5\left(\frac{|\mathbf{h}|}{a}\right)^3\right) & \text{si } |\mathbf{h}| \leq a \\ m & \text{si } |\mathbf{h}| > a \end{cases}, \tag{41.3}$$

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.

- **Exponencial.** Válido en $\mathbb{R}^d, d \geq 1$ y viene dado por:

$$\gamma(|\mathbf{h}|) = m\left(1 - \exp\left(-\frac{|\mathbf{h}|}{a}\right)\right). \tag{41.4}$$

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:

$$\gamma(|\mathbf{h}|) = m\left(1 - \exp\left(-\frac{|\mathbf{h}|^2}{a^2}\right)\right). \tag{41.5}$$

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:

$$\gamma(|\mathbf{h}|) = \begin{cases} m & \text{si } |\mathbf{h}| = 0 \\ 0 & \text{si } |\mathbf{h}| > 0 \end{cases}, \quad m > 0. \tag{41.6}$$

- **K-Bessel**. Válido in $\mathbb{R}^d, d \geq 1$. Su expresión es:

$$\gamma(|\mathbf{h}|) = m\left(1 - \frac{1}{2^{\alpha-1}\Gamma(\alpha)}\left(\frac{|\mathbf{h}|}{a}\right)^{\alpha} K_\alpha\left(\frac{|\mathbf{h}|}{a}\right)\right), \quad \alpha > 0, \tag{41.7}$$

  donde K_α es la función de segunda especie de orden α. Este modelo es que puede representar cualquier tipo de comportamiento cerca del origen. Por ejemplo, para $\alpha = 0,5$ se obtiene el modelo exponencial.

41.3.2.2. Semivariogramas con meseta y efecto de 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:

$$\gamma(|\mathbf{h}|) = m\left(1 - \left(\frac{2a}{|\mathbf{h}|}\right)^{\alpha} \Gamma(\alpha+1) J_{\alpha}\left(\frac{|\mathbf{h}|}{a}\right)\right), \tag{41.8}$$

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. 41.5 muestra una representación gráfica de los anteriores semivariogramas.

```
library(gstat)
show.vgms(models = c("Sph", "Exp", "Gau", "Nug", "Bes", "Wav"))
```

Figura 41.5: Representación de semivariogramas con meseta válidos (meseta y alcance efectivo unitarios; a excepción del efecto pepita puro)

41.3.2.3. 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:

$$\gamma(|\mathbf{h}|) = (|\mathbf{h}|)^{\alpha}, \quad \text{con } 0 < \alpha < 2, \tag{41.9}$$

- **Logarítmico**. Válido en $\mathbb{R}^d, d \geq 1$ y con expresión:

$$\gamma(|\mathbf{h}|) = b \log |\mathbf{h}| \;\text{ si }\; |\mathbf{h}| \geq 0, \tag{41.10}$$

  donde b es una constante.

Una representación gráfica de ambos semivariogramas puede verse en la Fig. 41.6.

```
library(gstat)
show.vgms(models = c("Pow", "Log"), sill = 1, range = c(2, 1), nugget = 0)
```

Figura 41.6: Representación de semivariogramas sin meseta válidos

41.3.3. 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 esacionariedead estricta y de segundo orden), y en $\mathbb{R}^d, d \geq 1$, se estiman (insesgadamente) los valores semivariográficos para un número determinado de distancias, por el método de los momentos:

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2\#N(\mathbf{h})} \sum_{N(\mathbf{h})} \left(Z(\mathbf{s}_i + \mathbf{h}) - Z(\mathbf{s}_i)\right)^2, \tag{41.11}$$

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 semivariograficos 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 diametro 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 numero de parejas para calcular el semivariograma, 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 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. 41.7 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 41.7: 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 Girona 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 (en geoR se divide en 13 porque los autores lo hicieron un viernes 13).

41.3.4. Ajuste semivariográfico

Cualquier funcion que dependa de una distancia y una dirección no es necesariamente un semivariograma, pues para ello tienen que cumplir los requisitos especificados en 41.3.1. 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). 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. 41.8 muestra el semivariograma empírico correspondiente a los puntos semivariográficos de la Fig. 41.7. 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, "SSErr")
#> [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: mínimos cuadrados ponderados con ponderaciones $\frac{N_{|\mathbf{h}|}}{|\mathbf{h}|^2}$, 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 utiliza la función por defecto: (i) alcance igual a 1/3 de la distancia máxima en la muestra; (ii) como efecto pepita se toma la

Figura 41.8: Semivariograma empírico. Temperaturas máximas (06/08/2022)

media de los tres primeros valores semivarigráficos; y (*iii*) como meseta parcial (meseta menos efecto pepita), la media de los cinco últimos valores semivariográficos.

41.4. 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 de un dominio D, y proporciona el mejor predictor lineal insesgado de la *v.r.* de interés en tales puntos o bloques no observados [7]. 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 deriva: la

[7] En 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).

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* 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 $\lambda_i, i = 1, 2, ..., n$, 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:

$$\begin{cases} \sum_{j=1}^{n} \lambda_j \gamma(\mathbf{s}_i - \mathbf{s}_j) + \alpha = \gamma(\mathbf{s}_i - \mathbf{s}_0), \qquad i = 1, ..., n \\ \sum_{i=1}^{n} \lambda_i = 1 \end{cases} \tag{41.12}$$

siendo la varianza del error de predicción: $\sigma^2_{OK}(\mathbf{s}_0) = \sum_{i=1}^{n} \lambda_i \gamma(\mathbf{s}_i - \mathbf{s}_0) + \alpha$, 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 <- krige(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, "º")
    },
    guide = guide_legend(reverse = TRUE, title = "Temp. max.")
  ) +
```

```
  theme_light() +
  theme(
    panel.background = element_blank(),
    panel.border = element_blank(),
    axis.title = element_blank(),
  )
```

Figura 41.9: Mapping de temperaturas máximas (06/08/2022).

El mapping de la Fig. 41.9 tiene poco valor si no se acompaña de otro que muestre la desviación típica de los errores de predicción.

```
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(),
  )
```

Figura 41.10: Desviaciones típicas del error de predicción

Como se aprecia en la Fig. 41.10 cuanto mayor es el número de localizaciones observadas alrededor del punto de predicción, menor es la desviación típica 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 42

Modelos econométricos espaciales

Andrés Vallone y Coro Chasco

42.1. La dependencia espacial

En muchas ocasiones los fenómenos sociales no son independientes del espacio geográfico en el cual se producen, esto se refleja en la primera ley de la Geografía enunciada por Tobler (1970a) "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 Minimos Cuadrados Ordinarios (MCO)

En este contexto, los MCO ya no son óptimos y, por tanto, los test t y F estadísticos pueden llevar a conclusiones erróneas(Anselin, 1988). Por eso es necesario encontrar la manera de incorporar el espacio geográfico en nuestros modelos de estimación. En este capítulo abordaremos este asunto, mostrando primero los métodos de exploración de datos espaciales, para luego mostrar las formas de modelización del espacio y los métodos de estimación.

Los modelos de econometría espacial se centrar en manejar las situaciones de dependencia espacial. Existe dependencia espacial cuando lo que sucede un una unidad espacial i esta influenciado por lo que sucede en una unidad espacial j y viceversa (Anselin, 2013). La dependencia espacial se traduce en que los valores de la variable de las unidades espaciales i y j con $i \neq j$ estan correlacionados entre sí, hecho que se conoce como autocorrelación espacial (Anselin, 2013). La autocorrelación espacial puede ser positiva, cuando la unidades espaciales con valores similares tienden a estar juntos (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 **??**. La ausencia de algún tipo de autocorrelación es lo que se entiende como aleatoriedad espacial (Anselin, 2013).

Figura 42.1: Patrones de autocorrelación espacial

42.1.1. Modelización del espacio: La matriz W

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 unidad de análisis y, por tanto, es crucial definir el vecindario para poder realizar los análisis.

La matriz de vecindad $\mathbf{W}_{n\times n}$ muestra la relación entre las n unidades espaciales analizadas. Esta matriz define la condición de vecindad y, de esta manera, la interacción existente entre las unidades espaciales. Esta matriz es simétrica y binaria, de forma que el elemento $w_{ij} = 1$ si las unidades espaciales i y j son vecinas y cero si no lo son. Por tanto, $w_{ii} = 0$ puesto que una unidad espacial no puede ser vecina de si misma. Existen distintas criterios de definición de esta matriz 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 **??** para difinir la matriz **W**

Figura 42.2: Criterios de vecindad extraido de [@martori2008incorporacion]

Las matrices **W** generadas bajo el criterio lineal consideran como vecinos a la unidad espacial i todas aquellas unidades que compartan un borde situadas en la misma dirección cardinal, norte sur o este oeste, de esta unidad. El resto de los criterios de contigüidad siguen los movimientos de las piezas del ajedrez para definir la vecindad de la unidad espacial i. La construcción de una matriz de vecindad bajo el criterio de la torre implica considerar como vecinos de la unidad

espacial i aquellas unidades espaciales situadas al norte, sur, este u oeste y que compartan un borde en común con dicha unidad. El uso del criterio de alfil considera como vecindad de la unidad i aquellas unidades situadas al noreste, noroeste, sureste o suroeste de la unidad espacial i y que tengan, al menos, un punto en común. El criterio de la reina considera como vecindario de la unidad espacial i alas unidades espaciales en todas las direcciones cardinales y que tengan al menos un punto en común con la unidad espacial i (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 unidad espacial i a los vecinos de los vecinos de dicha unidad. En este caso diríamos que la matriz es de orden 2 (los vecinos y los vecinos de los vecinos). Las matrices **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:

$$w_{ij}^{e} = \frac{w_{ij}}{\sum_{j=1}^{n} w_{ij}} \tag{42.1}$$

En palabras simples, se divide cada elemento de una fila de la matriz **W** por la suma de dicha fila. Esto asegura que cada elemento de la matriz **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 relación a la matriz **W** estaremos hablando de una matriz de pesos espaciales.

Para el cálculo de las matrices de vecindad utilizaremos el paquete `spedep` (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 ejemplo, utilizaremos el conjunto de datos del estudio de Guerry (1833) utilizados en [Anselin (2017)][1].

```
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
#> Percentage nonzero weights: 5.813149
#> Average number of links: 4.941176
#>
#> Weights style: W
```

[1] Una descripción completa de la base de datos está disponible en https://geodacenter.github.io/data-and-lab/Guerry/

```
#> 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 trabajamos con datos a nivel puntual, o cuando existen situaciones geográficas de no contigüidad como, por ejemplo, una isla, el uso de la matriz de contigüidad no es una condición 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 unidades espaciales 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 unidad espacial de tal manera que:

$$\begin{cases} w_{ij}(k) = 0 & i = j, \forall k \\ w_{ij}(k) = 1 & d_{ij} \leq d_i(k) \\ w_{ij}(k) = 0 & d_{ij} > d_i(k) \end{cases} \tag{42.2}$$

donde $d_i(k)$ es la k-esima menor distancia entre las unidades espaciales i y j. Utilizando funciones de Bivand (2022) y Pebesma (2022) y los datos de Vallone and Chasco (2020) calcularemos la matriz de 5 vecinos más próximos de las áreas urbanas chilenas con más de 2000 habitantes.

```
library(sf)
# Se extraen las coordenadas de los puntos
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 unidades espaciales, considerando vecinas a unidades espaciales que estén muy distantes entre ellas. Para solucionar esto podemos usar una configuración de vecindad basada en una distancia limite d_{max} de tal manera que:

$$\begin{cases} w_{ij} = 1 & d_{ij} \leq d_{max} \\ w_{ij} = 0 & d_{ij} > d_{max} \end{cases} \tag{42.3}$$

El problema de esta definición es la posibilidad de generar unidades espaciales aisladas debido a una definición muy estrecha de la distancia máxima (d_{max}). Esto se evita fijando la distancia

máxima 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 vencindad de distancia w_ij < d_max
k1 <- dnearneigh(coord, 0, distancia.critica)
w_dist <- nb2listw(k1)
```

Debe considerarse que la configuración de k vecinos y de la distancia censurada dan lugar a matrices binarias, mientras que las matrices $\mathbf{W}$ basadas en distancias, no lo son. Utilicemos el mismo conjunto de datos para realizar el cálculo de la matriz $\mathbf{W}$ basado en la distancia inversa. 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 unidad espacial j que se encuentre muy lejana a la unidad 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, donde d_{max} es fijada de tal manera de 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")
```

Hemos indicado que la matriz $\mathbf{W}$ se utiliza para capturar los efectos del espacio a partir de medias ponderadas, es decir mediante la matriz $\mathbf{W}$ seremos capaces de construir el *retardo* espacial. El retardo espacial $\mathbf{W}y$ de la variable 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 la variable y en el vecindario de cada unidad espacial i,

42.2. Medidas de Autocorrelación

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 rezago espacial en una gráfico de puntos. La Figura @ref{fig:moranplot} muestra un diagrama de Moran, la linea horizontal intercortada muestra

el promedio del rezago espacial, la linea vertical intercortada muestra el promedio de la variable clergy. El dividir el diagrama a partir de las medias permite generar cuatro zonas en el diagrama, el área "HH" que contiene a las unidades espaciales cuyo valor de la variable clergy es superior al promedio y su vecindario tiene valores de dicha variable superiores al promedio. Las unidades espaciales que se sitúen en el área "LL" presenta valores de la variable cleargy menores al promedio y su entorno tienen valores inferiores al promedio. El área "LH" contendrá a las unidades espaciales cuyo valor de la variable cleargy es inferior al promedio y su vecindario tiene valores de cleargy superiores al promedio y lo inverso sucede en el área "HL".

A partir del diagrama es posible observar la situación de una variable respecto a su entorno, si las unidades espaciales se sitúan mayoritariamente en las zonas "HH" y "LL" significa que la unidades espaciales con altos valores (superiores al promedio) de la variable de interés están rodeadas por unidades espaciales con altos valores de dicha variable, y las unidades espaciales con valores bajos (inferiores al promedio) están rodeadas de unidades espaciales con valores bajos, lo cual es una señal de existencia de autocorrelación espacial positiva. Si la concentración se realiza en las áreas "HL" y "LH" es indicativo de autocorrelación espacial negativa.

```
w_reina_francia <- poly2nb(guerry, queen = TRUE) |> nb2listw()
## Diagrama de Moran
moran.plot(guerry$clergy, w_reina_francia,
  xlab = "Clergy",
  ylab = "Rezago espacial de Clergy"
)
```

Figura 42.3: Diagrama de Moran de la variable Clergy

La Figura 42.3 da indicios de la existencia de autocorrelación positiva, es decir que las provincias tienden a estar rodeadas de provincias con cantidades similares de clérigos, altos valores junto a altos valores y bajo valores junto a bajos valores. El diagrama de Moran es una herramienta gráfica, para poder comprobar estadísticamente la existencia de autocorrelación global se utilizará el indicador I de Moran

42.2.1. El indicador I de Moran

Si definimos $x = y - \bar{y}$, el indicador de Moran se calcula como:

$$I = \frac{n}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} x_i w_{ij} x_j}{\sum_{i=1}^{n} x_i^2} \tag{42.4}$$

Este indicador se encuentra en el rango $[-1, 1]$ y el signo del indicador coincide con el tipo de autocorrelación: valores positivos son indicativos de autocorrelación positiva y valores negativos son indicadores de la existencia de autocorrelación negativa. Observando detenidamente la ecuación (42.4) es posible detectar que el cálculo del indicador es calculado como la ratio de la covarianza de la variable Y con su retardo espacial y la varianza de la variable y. Por tanto, el indicador coincide con el coeficiente de una regresión lineal de $\mathbf{W}y$ sobre y. La linea con pendiente positiva presente en 42.3 es precisamente el resultado de la regresión lineal entre $\mathbf{W}y$ e 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. Es posible calcular la significación estadística del indicador mediante un proceso de aleatoriedad espacial, considerando como hipótesis nula la inexistencia de autocorrelación espacial.(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 homogeneidad espacial no pudiendo rechazar la existencia de autocorrelación positiva.

Es posible detectar agrupaciones espaciales a partir de los indicadores de autocorrelación local LISA (Anselin, 1995). Los métodos LISA descomponen los indicadores globales y verifican en cuánto contribuye cada unidad espacial a la formación del valor general, permitiendo obtener un valor de significación para cada *cluster* formado por los valores similares de cada unidad espacial y sus vecinos(Chasco Yrigoyen, 2006). Es posible medir la significación de cada unidad en particular; por ello es posible que los indicadores globales no indiquen la presencia de autocorrelación espacial y aun así encontrar clústeres estadísticamente significativos.

42.3. Modelos econométricos espaciales de corte transversal

Los modelos espaciales deben ser identificados, antes de proceder a su estimación, inferencia y contraste. Para ello, es importante disponer de una estrategia de identificación propia, que permita al investigador conocer los parámetros poblacionales correctos 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, o 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, podemos 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) que tiene en cuenta las buenas propiedades de las propuestas anteriores.

Según se presenta en la Figura 42.4, la estrategia híbrida comienza con la estimación de un modelo básico sin efectos espaciales:

$$y = \alpha\iota_n + \mathbf{X}\beta + \epsilon \tag{42.5}$$

siendo y el vector de la variable dependiente, de orden $(n \times 1)$; $\mathbf{X}$ la matriz de variables explicativas, de orden $(n \times k)$; ι_n un vector formado por unos, de orden $(n \times 1)$; α, β son el grupo de $(k+1)$ parámetros a estimar; y ϵ es la variable de perturbaciones aleatorias, de orden $(n \times 1)$, que se distribuye como $\epsilon \sim N(0, \sigma_\epsilon \mathbf{I}_n)$, siendo $\mathbf{I}_n$ la matriz identidad de orden n.

Este modelo se estima por mínimos cuadrados ordinarios (MCO) y luego, se calculan los tests LM del Multiplicador de Lagrange sobre la variable de los errores de la regresión para contrastar si son ruido blanco desde el punto de vista espacial. Esta hipótesis básica se rechaza en cuanto que alguno de dichos tests, que se distribuyen como una Ji cuadrado con 1 grado de libertad (χ^2_1), resulte estadísticamente significativo. Se trata de dos tests orientados hacia una única hipótesis alternativa: el LMLAG, Multiplicador de Lagrange para la hipótesis de variable dependiente espacialmente retardada, y el LMERR, para la hipótesis de dependencia residual.

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):

$$y = \rho\mathbf{W}y + \alpha\iota_n + \mathbf{X}\beta + \mathbf{WX}\theta + \epsilon \tag{42.6}$$

siendo ρ un parámetro y θ un vector de k parámetros autorregresivos espaciales.

Este modelo general incluye dos tipos de interacción espacial, los efectos endógenos ($\mathbf{W}y$) y exógenos ($\mathbf{WX}$). La variable endógena espacialmente retardada ($\mathbf{W}y$) referida al mismo momento del tiempo que la variable dependiente (y) produce en los estimadores MCO una situación

de simultaneidad y, por tanto, sesgo, ineficiencia e inconsistencia. Por eso, se recomienda su estimación por el método de máxima verosimilitud (MV), que supone normalidad en la distribución de los errores (ver Anselin (1988), cap. 6).

La estimación MV de este modelo permite utilizar la ratio de verosimilitud (LR), cuya distribución sigue una Ji al cuadrado con k grados de libertad, como test para contrastar las hipótesis nulas $H_0(\theta = 0)$ y $H_0(\rho = 0)$. En este punto, se pueden dar tres casos:

1) Si se acepta la primera hipótesis, pero no la segunda, siempre y cuando el test LMLAG > LMERR, el SDM debería simplificarse a un modelo del retardo espacial o modelo autorregresivo espacial de orden 1 (SAR):

$$y = \rho \mathbf{W} y + \alpha \iota_n + \mathbf{X}\beta + \epsilon \tag{42.7}$$

Este modelo se estima por MV si los errores de la estimación por MCO se distribuyen como una normal.

2) Si se acepta la segunda hipótesis, pero no la primera, y LMERR > LMLAG, debería seleccionarse el modelo del error espacial (SEM):

$$\begin{cases} y = \alpha \iota_n + \mathbf{X}\beta + u \\ u = \lambda \mathbf{W} \epsilon + \epsilon \end{cases} \tag{42.8}$$

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 modelo espacial regresivo transversal (SLX):

$$y = \alpha \iota_n + \mathbf{X}\beta + \mathbf{W}\mathbf{X}\theta + \epsilon \tag{42.9}$$

Este modelo puede estimarse por el método MCO ya que, si las variables explicativas son exógenas, también lo serán sus correspondientes retardos espaciales. La estimación puede considerar todas las variables exógenas espacialmente retardadas o un subconjunto de ellas, para contrastar la hipótesis nula $H_0(\theta = 0)$. Si esta hipótesis no pudiera ser 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 $H_0(\theta = 0)$ tuviera que rechazarse, habría que estimar el modelo SDM con las variables **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 si no pudiera rechazarse, 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 42.4: Estrategia de especificación híbrida (Elhorst, 2010)

El siguiente conjunto de secuencias de código muestran como estimar los modelos que intervienen en la estrategia de modelización de Elhorst. Para ello, se utilizan un conjunto de datos de los municipios (NUTS 5) que forman parte de las áreas urbanas de España utilizados en Mella and Chasco (2006).

```
library(spatialreg)
library(tseries)

# Matriz de pesos espaciales
coord <- coordinates(gdpmap)
k6 <- knearneigh(coord, k = 6)
```

```
dmins <- knn2nb(k6) |> nb2listw(style = "W")
# Estimación modelo básico por MCO
gdpols <- lm(LPGH ~ LGH85 + BANK + UNI01 + PAT00, data = gdpmap)
summary(gdpols)
jarque.bera.test(gdpols$res) # Test normalidad de residuos
lm.LMtests(gdpols, dmins, test = "all") # Grupo de tests LM

# Estimación modelo SDM por MV
gdpsdm <- lagsarlm(LPGH ~ LGH85 + BANK + UNI01 + PAT00,
  data = gdpmap,
  listw = dmins, type = "mixed"
)
summary(gdpsdm)

# Estimación modeelo SAR por MV
gdpsar <- lagsarlm(LPGH ~ LGH85 + BANK + UNI01 + PAT00,
  data = gdpmap,
  listw = dmins
)
summary(gdpsar)
LR.Sarlm(gdpsdm, gdpsar) # Test LR: SDM vs. SAR

# Estimación del modelo SEM por MV
gdperr <- errorsarlm(LPGH ~ LGH85 + BANK + UNI01 + PAT00,
  data = gdpmap,
  listw = dmins, tol.solve = 1e-16
)
summary(gdperr)
LR.Sarlm(gdpsdm, gdperr) # 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)

gdpslx <- lm(LPGH ~ LGH85 + BANK + UNI01 + PAT00 + WLGH85 + WBANK + WUNI01 +
↪  WPAT00, data = gdpmap)
summary(gdpslx) # Modelo SLX completo

gdpslx2 <- lm(LPGH ~ LGH85 + BANK + UNI01 + PAT00 + WLGH85 + WPAT00, data =
↪  gdpmap)
summary(gdpslx2) # Modelo SLX parsimonioso
```

```
LR.Sarlm(gdpsdm, gdpslx2) # Test LR: SDM vs. SLX
```

42.3.1. Estimación SAR

A continuación se muerta la salida de la estimación del modelo SAR, donde puede notarse que todos los parámetros estimados, incluido el parámetro ρ, son estadísticamente significativos.

```
#>
#> Call:lagsarlm(formula = LPGH ~ LGH85 + BANK + UNI01 + PAT00, data = gdpmap,
#>     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
```

42.3.2. Comparando SAR contra SDM

A continuación se muestra el resultado el LR test comparando la estimación del modelo SDM contra el modelo SAR, observando el valor del p-value de la prueba es posible rechazar que el valor de los parámetros restringidos sea cero y por tanto el modelo SDM es más adecuado para explicar este fenómeno 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
```

42.3.3. Interpretación de los estimadores de los modelos de autocorrelación espacial

Sólo en los modelos de autocorrelación espacial en los que el efecto endógeno ($\mathbf{W}y$) 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, porque la forma reducida coincide con la forma estructural. Es decir, el efecto marginal de un cambio del valor de una variable explicativa continua para una determinada localización coincide con el valor del estimador correspondiente a dicha variable.

$$\frac{\partial y_i}{\partial x_{i,k}} = \beta_k \tag{42.10}$$

En los modelos SAR y SDM, la correcta interpretación de los estimadores implica antes pasar de la forma estructural a la forma reducida. Así, por ejemplo, en el modelo SAR de la expresión (42.7) la forma reducida sería (bajo ciertas condiciones de invertibilidad):

$$y = (\mathbf{I} - \rho\mathbf{W})^{-1}(\alpha\iota_N + \mathbf{X}\beta + \epsilon) \tag{42.11}$$

El término $(\mathbf{I} - \rho\mathbf{W})^{-1}$ es denominado multiplicador espacial y, utilizando la expansión potencial, puede expresarse también del modo siguiente:

$$(\mathbf{I} - \rho\mathbf{W})^{-1} = \mathbf{I} + \rho\mathbf{W} + \rho^2\mathbf{W}^2 + \ldots \tag{42.12}$$

Si se utiliza esta nueva expresión en la ecuación (42.11) para la media condicional, tenemos que el valor de 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 $\rho\mathbf{W}\mathbf{X}\beta$), del valor de las explicativas en las localizaciones vecinas a las vecinas (a través del término $\rho^2\mathbf{W}^2\mathbf{X}\beta$), etc.

$$E(y|\mathbf{X}) = \mathbf{X}\beta + \rho\mathbf{W}\mathbf{X}\beta + \rho^2\mathbf{W}^2\mathbf{X}\beta + +\rho^3\mathbf{W}^3\mathbf{X}\beta + \ldots \tag{42.13}$$

LeSage and Pace (2009) presentan los efectos causados por el cambio en el valor de una localización, j, de la variable x_k sobre otra localización, i, de la variable y como una matriz

completa, $\mathbf{S}_k(\mathbf{W})_{ij}$, de orden $n \times n$. Así, cada variable explicativa x_k del modelo tendrá una matriz completa propia de impactos sobre la variable dependiente.

$$\mathbf{S}_k(\mathbf{W})_{ij} = \frac{\partial y_i}{\partial x'_k} = (\mathbf{I_n} - \rho \mathbf{w})^{-}1\beta_k \tag{42.14}$$

En este caso, también podemos distinguir efectos directos e indirectos. El efecto directo sería el efecto causado por cambios en el valor de x_k, en una localización i sobre el valor de y en esa misma localización i. Estos efectos constituyen los valores de la diagonal principal de la matriz $\mathbf{S}_k(\mathbf{W})_{ii}$. El efecto indirecto está constituido 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 x_k, en una localización j afecta al valor de 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 observación n y luego volver de nuevo al punto de partida.

Por ejemplo, el valor de $\mathbf{S}_k(\mathbf{W})_{11}$ significaría el efecto directo de un cambio en el valor de la variable x_k en la localización 1 (x_{k1}) sobre el valor de la variable y en esa misma localización 1 (y_1), mientras que el valor de $\mathbf{S}_k(\mathbf{W})_{12}$ sería el efecto indirecto de un cambio en el valor de la variable x_{k1} sobre el valor de la variable y en la localización 2 (y_2). En las filas, la matriz $\mathbf{S}_k(\mathbf{W})_{ij}$ tiene los efectos de x_k desde cada localización i "hacia" todas y cada una de las localizaciones j, mientras que las columnas representan el efecto sobre cada localización j "desde" todas y cada una de las localizaciones i.

Dado que no es posible construir un proceso inferencial para todos los impactos de la matriz $\mathbf{S}_k(\mathbf{W})_{ij}$ LeSage y Pace proponen realizar la inferencia sobre el valor medio de los efectos directos y totales, extrayendo por diferencia los efectos indirectos:

$$\bar{M}(k)_{directo} = tr(\mathbf{S}_k(\mathbf{W}))/n \tag{42.15}$$

$$\bar{M}(k)_{total} = \iota'_{\mathbf{n}}\mathbf{S}_k(\mathbf{W})\iota_{\mathbf{n}}/n \tag{42.16}$$

$$\bar{M}(k)_{indirecto} = \bar{M}(k)_{total} - \bar{M}(k)_{directo} \tag{42.17}$$

El siguiente conjunto de secuencias presentan el cálculo de las matrices de efectos y la inferencia para los modelos SAR y SDM correspondientes al ejemplo del modelo anteriormente 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
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")
```

42.3.4. Impacto del SAR

A continuación se presenta la salida del cálculo de los efectos para el modelo SAR (LeSage and Pace, 2009) donde puede notarse que todos los impactos son estadísticamente significativos y salvo la variable LGH85 todos tienen efectos positivos sobre la variable LPGH

```
#> Impact measures (lag, trace):
#>              Direct      Indirect         Total
#> LGH85 -3.361200e-02 -1.833440e-02 -0.0519463985
#> BANK   6.780328e-05  3.698479e-05  0.0001047881
#> UNI01  4.780827e-04  2.607807e-04  0.0007388635
```

```
#> PAT00  4.818749e-02  2.628492e-02  0.0744724116
#> ==========================================================
#> Simulation results ( variance matrix):
#> ==========================================================
#> Simulated standard errors
#>             Direct     Indirect        Total
#> LGH85 4.330382e-03 5.661226e-03 7.112130e-03
#> BANK  1.299226e-05 1.470493e-05 2.376829e-05
#> UNI01 8.505222e-05 9.339449e-05 1.461224e-04
#> PAT00 1.833567e-02 1.535729e-02 3.175518e-02
#>
#> Simulated z-values:
#>          Direct  Indirect     Total
#> LGH85 -7.731876 -3.314485 -7.346044
#> BANK   5.181691  2.604510  4.443778
#> UNI01  5.651099  2.901134  5.143554
#> PAT00  2.623368  1.821494  2.395654
#>
#> Simulated p-values:
#>       Direct     Indirect   Total
#> LGH85 1.0658e-14 0.00091812 2.0406e-13
#> BANK  2.1988e-07 0.00920058 8.8393e-06
#> UNI01 1.5943e-08 0.00371815 2.6959e-07
#> PAT00 0.0087065  0.06853186 0.016591
```

RESUMEN En este capítulo se introduce a la moderación del espacio en la estimación econométrica. Primero, observamos la heterogeneidad espacial de los datos a partir de los mapas temáticos y el uso de indicador de autocorrelación espacial de Moran. Luego, construimos la matriz de pesos espaciales bajo distintas especificaciones. Y, por último, se muestra la taxonomía de los modelos econométricos espaciales presentado la estrategia de especificación híbrida y la interpretación de los coeficientes estimados

Capítulo 43

Procesos de punto

Jorge Mateu y Mehdi Moradi

43.1. Spatial point patterns on $\mathbb{R}^2$

Let $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}, 0 \leq n < \infty$, be an observed realisation of a simple (i.e no multiple events per location) and finite point process X on $\mathbb{R}^2$ within the window $W \subset \mathbb{R}^2$ and with the associated distance metric $d(u, v)$. For any arbitrary set $A \subset \mathbb{R}^2$, the cardinality of X is governed by the count function

$$N(X \cap A) = \sum_{x \in X} \mathbf{1}\{x \in A\} < \infty.$$

Moreover, according to the Campbell formula (Baddeley et al., 2015), for any measurable function $f : \mathbb{R}^2 \to [0, \infty)$ we have

$$\mathbb{E}\left[\sum_{x \in X} f(x)\right] = \int_{\mathbb{R}^2} f(u)\lambda(u)\mathrm{d}u,$$

where $\lambda(\cdot)$ is called the intensity function of X, and governs its spatial distribution. In fact, $\lambda(u)$ gives the expected number of points per unit area in the vicinity of $u \in \mathbb{R}^2$. Letting $f(x) = \mathbf{1}\{x \in A\}$, we can easily see the relationship between the intensity function $\lambda(\cdot)$ and the count function N as

$$\mathbb{E}\left[N(X \cap A)\right] = \int_A \lambda(u)\mathrm{d}u.$$

If the intensity function $\lambda(\cdot)$ is constant, i.e. $\lambda(\cdot) = \lambda$, the process X is called homogeneous, otherwise it is called inhomogeneous, meaning that its spatial distribution varies over its state

space. For further details we encourage the interested reader to see Møller and Waagepetersen (2003); Illian et al. (2008); Diggle (2013); Baddeley et al. (2015).

In practice one often observes a single realisation, thus it is of great importance to have an estimate of $\lambda(\cdot)$ that can mimic the spatial distribution of the underlying process, which is supposed to have generated the observed pattern. Hence, we next go through different non-parametric intensity estimators.

43.1.1. Kernel-based intensity estimation

The two most frequently used non-parametric kernel-based intensity estimators for spatial point patterns on $\mathbb{R}^2$ are

$$\widehat{\lambda}_{\sigma}^{\mathrm{U}}(u) = \frac{1}{c_{\sigma,W}(u)} \sum_{i=1}^{n} \kappa_{\sigma}(u - x_i), \quad u \in W, \tag{43.1}$$

and

$$\widehat{\lambda}_{\sigma}^{\mathrm{JD}}(u) = \sum_{i=1}^{n} \frac{\kappa_{\sigma}(u - x_i)}{c_{\sigma,W}(x_i)}, \quad u \in W, \tag{43.2}$$

where κ_{σ} is a probability density function on $\mathbb{R}^2$ with bandwidth σ, and

$$c_{\sigma,W}(u) = \int_W \kappa_{\sigma}(u - v)\mathrm{d}v, \quad u \in W, \tag{43.3}$$

is the mass of the kernel centred at $u \in W$, and plays the role of an edge corrector to compensate for lack of information outside W. Note that we, in practice, only see a realisation of X within a bounded/limited domain W. Regardless of the choice σ, the estimator (43.1) is pointwise unbiased if the true intensity function is constant (Diggle, 1985), while the estimator (43.2) conserves the total mass (Jones, 1993). We note that the estimators (43.1) and (43.2) are often called `uniformly-edge-corrected` and `Jones-Diggle` (Rakshit et al., 2019). Throughout, we consider the well-known Gaussian kernel function (Silverman, 1986, Page. 42)

Regarding practical concerns, the performance of kernel-based intensity estimators strongly depends on the bandwidth parameter, so that a small bandwidth leads to an under-smoothed estimate (low bias and high variance) whereas a large bandwidth results in an over-smoothed estimate (high bias and low variance). For a given point pattern **x**, the estimators (43.1) and (43.2) can be calculated using the function density.ppp from spatstat.core by specifying diggle=FALSE and diggle=TRUE, respectively.

43.1.1.1. Bandwidth selection

Scott (1992) proposed to choose the bandwidth through the rule of thumb

$$(s_x n^{-1/6}, s_y n^{-1/6}),$$

for each of the x, y Cartesian coordinates, in which s_x, s_y stand for the sample standard deviation of the x, y coordinate values of data locations. Despite being a rule of thumb, this procedure

can be quite useful for exploratory analysis. The function bw.scott from spatstat.core calculates such rule of thumb.

Cronie and Van Lieshout (2018) proposed to find an optimum bandwidth by minimising

$$CvL(\sigma) = \left(|W| - \sum_{i=1}^{n} 1/\widehat{\lambda}^*_\sigma(x_i) \right)^2,$$

where $\widehat{\lambda}^*_\sigma(x_i)$ is a non-corrected intensity estimate (either of (43.1) or (43.2) without the correction term) at x_i using the bandwidth σ. The idea behind this proposal lies on the fact that, using Campbell formula, we have

$$\mathbb{E}\left[\sum_{x \in X} 1/\lambda(x) \right] = \int_W (1/\lambda(x))\lambda(x)\mathrm{d}u = |W|.$$

For a given point pattern $\mathbf{x}$, the function bw.CvL from spatstat.core calculates the bandwidth using Cronie and van Lieshout's criterion.

43.1.2. Practical examples

We next employ the above-mentioned intensity estimators and bandwidth selection approaches to study the spatial distribution of two datasets: Wildfire in Nepal (Figure **??**), and Crime in Medellín, Colombia (Figure **??**). Throughout this section, we make use of the R packages spatstat, version 2.3-0, (Baddeley and Turner, 2005; Baddeley et al., 2015) for point pattern analysis and raster, version 3.5-15, (Hijmans, 2022) for graphical representations. We add that the package spatstat is recently divided into a family of sub-packages spatstat.utils, spatstat.data, spatstat.sparse, spatstat.geom, spatstat.random, spatstat.core, spatstat.linnet, so that spatstat itself acts as an umbrella package. We warn the readers to be aware of future changes of spatstat to comply with CRAN's suggestions concerning package size.

43.1.2.1. Example 1: Wildfires in Nepal

By courtesy of Ganesh Prasad Sigdel, we present here the following data for Nepal which contains (forest) wildfires in Nepal as geo-referenced point data during 2016, obtained from ICIMOD-Nepal. During the studied period Nepal experienced 5757 forest fires in which 475 of them happened in Surkhet District that is a district in Karnali Province of mid-western Nepal. We below begin with loading the essential R packages for this section.

```
library(spatstat)
library(raster)
```

Below we estimate the smoothing bandwidth parameter for the considered forest fire data using the reviewed methods in Section **??**. Scott's rule gives (50253.47 m, 21158.42 m) and Cronie

and van Lieshout's criterion (CvL) estimates the bandwidth as 36513.16 m. We note that one may wish to search for an optimum bandwidth over a finer grid of possible values. This can currently be controlled by arguments ns for bw.CvL.

```
data(nepal)
scott_nepal <- bw.scott(nepal) # Scott's rule
CvL_nepal <- bw.CvL(nepal) # Cronie and van Lieshout's criterio
```

Having estimated bandwidth through different methods, we next estimate the intensity of the wildfire data by employing the intensity estimators (43.1) and (43.2). The function density.ppp provides kernel-based intensity estimation for point patterns on $\mathbb{R}^2$, noting that as default it currently makes use of the uniformly-edge-corrected estimator (43.1) with the Gaussian kernel function. We set leaveoneout=FALSE to not calculate the so-called leave-one-out estimator, and positive=TRUE to force density values to be positive. We note that due to numerical errors inherent in the Fast Fourier Transform (finite-precision arithmetic) we might get quite small negative values in very sparse areas, see the help page of 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)
```

Now we turn to estimate the intensity through the Jones-Diggle estimator (43.1) by setting diggle=TRUE in 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)
```

Having obtained different intensity estimates after considering distinct bandwidth selection approaches, we next aim to show the obtained estimates and discuss their possible discrepancies. For a better graphical representation, we convert the obtained intensity images from class im to objects of class raster and then merge them into a RasterStack. Figure 43.1 shows different intensity estimates in which a high intensity is assigned to the south and south-west of Nepal highlighting a non-uniform distribution for the forest fire data. The slight difference between the maps generated by the Scott's rule and CvL is that, unlike CvL, Scott's rule further discloses an area of higher intensity in the south-west.

```
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
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))
```

Figura 43.1: Kernel-based intensity estimation for the wildfire data in Nepal during 2016. Layer's names start with bandwidth selection approach followed by the employed kernel and the edge-correction. Intensity values are forest fires per ten thousand km.

43.1.2.2. Example 2: Crimes in Medellín

Medellín is the second most populated city in Colombia[1] with an urban territory enclosing an area of roughly 105km^2. Medellín has suffered with crime for many years, being known as home of dangerous criminals. In 2018 Secretariat of Security of Medellín reported that 40 % of the citizens felt insecure, having delivered about 20607 theft complaints [2]. In addition, the police department recognises the need of hiring almost 2000 more policemen to fight against homicide, theft and micro-traffic[3].

[1] DANE (2019) Proyecciones de población departamentales y municipales por área 2005–2020. www.dane.gov.co.

[2] Restrepo V (2019) ¿qué tan segura se siente la gente en medellín? El Colombiano.

[3] Monsalve C (2019) Medellín necesita 2000 uniformados más para reforzar seguridad. Blu Radio.

In this chapter, we only analyse the spatial distribution of the locations of crimes occurred in Medellín during 2005 (Sanabria et al., 2022). During 2005, 910 crimes are committed for which the percentage of men being victims is 66 %, 28 % of crimes correspond to weekends, the percentage of hold-up theft is 42 % being the most common type, and the percentage of victims having an age between 20 and 40 increases up to 60 %.

We point out that this dataset contains types of crimes which do not necessarily happen over the street network of Medellín such as card cloning, hoax, scopolamine-related, etc. Thus, we consider this dataset as a point pattern on $\mathbb{R}^2$. One could, however, make a classification of crime types.

```
data(medellin)
scott_med <- bw.scott(medellin) # Scott's rule
CvL_med <- bw.CvL(medellin) # Cronie and van Lieshout's criterio
```

The Scott's rule estimates the bandwidth as (691.31m, 954.20m) and CvL leads to 692.31m. Then, similar to the case of wildfires in Nepal, we make use of the function density.ppp to compute the intensity estimators (43.1) and (43.2) under the same settings as Section ??.

```
# uniformly-edge-corrected estimator and Gaussian kernel
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)

# Jones-Diggle estimator and Gaussian kernel
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)
```

Figure 43.2 shows the estimated intensities using different choices of bandwidth selection methods. At a glance, we can generally see a non-uniform distribution for crimes in Medellín. Regardless of the choice of methodology for choosing bandwidth, the estimates highlight two big hotspots in the central area of Medellín, possibly with different radius, slight difference in other areas. Similar to the case of wildfires in Nepal, one can see that the effect of edge-corrections are 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 43.2: Kernel-based intensity estimation for the crime data in Medellín, Colombia, during 2005. Layer's names start with bandwidth selection approach followed by the employed kernel and the edge-correction. Intensity values are crime per haundred km.

43.1.3. Kernel-based intensity estimation over irregular domains

The estimators (43.1) and (43.2) may suffer from important issues such as tunnelling the mass, bias near the boundary, simultaneous over- and under-smoothing artefacts, leading to physically implausible results under certain situations (Baddeley et al., 2022). These problems are more noticeable when data are observed over an irregular domain. As a remedy Baddeley et al. (2022) proposed to estimate the intensity via the heat kernel which can be defined as the transition probability density of a Brownian motion on W that respects a boundary. Indeed, their proposal, which is called the diffusion estimator, is of the form

$$\widehat{\lambda}_t(u) = \sum_{i=1}^{n} \kappa_t(u|x_i), \tag{43.4}$$

where $t = \sigma^2$ (σ being the smoothing bandwidth in (43.1) and (43.2)) and $\kappa_t(\cdot|x_i)$ is the heat kernel. This estimator is both unbiased, under homogeneity, and mass-conserving simultaneously. Baddeley et al. (2022) further extended some of the bandwidth selection approaches to their proposal including the likelihood cross validation and Cronie–Van Lieshout criterion. The diffusion estimator above is currently accessible through the function densityHeat.ppp, the versions of Cronie and Van Lieshout criterion and likelihood cross validation to find an optimum bandwidth are accommodated in the functions bw.CvLHeat and bw.pplHeat. All such functions belong to spatstat.core.

Below we employ the diffusion estimator to briefly see its performance compared to that of (43.1) through an application of active fires within USA and Central America (discarding the islands) from 24th February to 3rd March 2022[4]. Active fires refer to Near Real-Time (NRT) fire/thermal anomaly data which represents the centres of some 1km pixels that are flagged by

[4] https://firms.modaps.eosdis.nasa.gov/active_fire/

an algorithm as containing one or more fires. The locations do not necessarily confirm fires, but rather susceptible pixels for which the difference between the land cover around them and a potential fire meets a given threshold.

The smoothing bandwidths for the computation of both estimators (43.1) and (43.4) are chosen by the Cronie and Van Lieshout criterion. Here, due to dealing with a quite bigger area compared to our previous examples, we consider ns=50, meaning that a vector of size 50 is used to search for an optimum bandwidth (default is 16), and dimyx=512 to get intensity images with better resolution (default is an image of size 128×128). The chosen bandwidths to compute (43.1) and (43.4) are approximately 556.3km and 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(activefires, sigma =
↪  heat_CvL_northcentre, leaveoneout = FALSE, dimyx = 512)
```

Both estimates are merged into a RasterBrick object which is represented in Figure 43.3. The domain in this example is not regular; the Florida state, (south) lower California state, and all Central America makes the studied region quite irregular. In such a situation, it might be quite an implausible result if our estimate transfers kernel mass across the gulf of California/Mexico between two halves. The intensity map on the left-hand side of Figure 43.3 shows that the uniformly-edge-corrected estimate suffers from tunnelling the mass across the region. However, the intensity map on the right-hand side, obtained by the diffusion estimator, shows quite less mass migration, and seems to be more realistic if one takes into account the spatial distribution of points.

```
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

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 43.3: Kernel-based intensity estimation for the active fires within USA and Central America (discarding the islands) from 24th February to 3rd March 2022. Left: uniformly-edge-corrected estimate with Gaussian kernel, Right: diffusion estimate. Equivalently optimum bandiwdth for both methods are chosen by the Cronie and Van Lieshout criterion. Intensity values are fires per thousand km

43.1.4. Voronoi-based intensity estimators

As previously noted, the performance of kernel-based intensity estimators strongly depends on the smoothing parameter bandwidth, and even more, so it might be quite hard (if not impossible) to find a single fixed bandwidth which smooths the intensity equally fine over the spatial domain W when there are abrupt changes in the spatial distribution of points. A spatially-varying (adaptive) bandwidth was then suggested to overcome such issues, but at the cost of increased complexity (Davies and Baddeley, 2018; Baddeley et al., 2022). As an alternative approach, and to avoid the need for bandwidth parameter selection in advance, one can employ Voronoi-based intensity estimators which are fully non-parametric (Barr and Schoenberg, 2010).

For any $x \in \mathbf{x}$, its associated Voronoi/Dirichlet cell $\mathcal{V}_x$ that consists of all $u \in W$ which are closer to x than any other $y \in \mathbf{x} \setminus \{x\}$, is given by

$$\mathcal{V}_x = \{u \in W : d(x,u) \leq d(y,u) \text{ for all } y \in \mathbf{x} \setminus \{x\}\}. \tag{43.5}$$

The Voronoi-based intensity estimator, at any arbitrary point $u \in W$, is then of the form

$$\widehat{\lambda}^V(u) = \sum_{x \in \mathbf{x}} \frac{\mathbf{1}\{u \in \mathcal{V}_x\}}{|\mathcal{V}_x|}. \tag{43.6}$$

The estimator $\widehat{\lambda}^V(u)$ conserves mass (similar to $\widehat{\lambda}^{\mathrm{JD}}_\sigma(u)$), and is simultaneously unbiased when the true intensity is constant (similar to $\widehat{\lambda}^{\mathrm{U}}_\sigma(u)$) which are the properties the diffusion estimator holds as well. However, Moradi et al. (2019) showed that $\widehat{\lambda}^V(u)$ suffers from a quite high variance

which might be indeed a result of under-smoothness in very dense areas and over-smoothness in sparse regions. Thereafter, they proposed to overcome such smoothness issues of $\widehat{\lambda}^V(u)$ by sub-sampling some $m \geq 1$ rescaled copies of $\mathbf{x}$ through independent p-thinning. In fact, they proposed to estimate the intensity function through

$$\widehat{\lambda}^V_{p,m}(u) = \frac{1}{m}\sum_{i=1}^{m} \frac{\widehat{\lambda}^V_i(u)}{p}, \ u \in W, \tag{43.7}$$

where $\widehat{\lambda}^V_i(u)$ corresponds to the Voronoi-based intensity estimator of the i-th thinned pattern. The estimator $\widehat{\lambda}^V_{p,m}(u)$ is called resample-smoothed, and, in addition to the statistical properties of $\widehat{\lambda}^V(u)$, holds some other favourable statistical properties including a very low variance. Employing such an estimator means that one again needs to deal with parameter selection (m, p) in advance; however Moradi et al. (2019) proposed both a rule-of-thumb ($m = 400$ and $p \leq 0{,}2$) and a data-driven cross-validation approach. Both estimators (43.6) and (43.7) are accessible through the function densityVoronoi.ppp from spatstat.core in which arguments f and nrep control the probability p and number of thinnings m, respectively. We note that by setting f=1 one can obtain an estimate based on the voronoi estimator (43.6).

Next, we estimate the intensity of the wildfire data in Nepal (Section **??**) using the resample-smoothed Voronoi estimator (43.7) by considering different choices for the retention probability p.

```
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)
```

The obtained estimates, which similarly to the outcomes of density.ppp are of class im, are then merged into a RasterBrick object and are prepared for graphical representation.

```
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)
```

Figure 43.4 shows the voronoi-based estimated intensities for the wildfire data in Nepal based on a sequence of retention probabilities. An under-smoothness and high variance for large probabilities is visible. One can also see that retention probabilities less than or equal to 0.2 give rise to smoothed heat-maps that can reflect local variations better than fixed-bandwidth kernel estimators.

Figura 43.4: Resample-smoothed Voronoi intensity estimation for the wildfire data in Nepal during 2016 based on different retention probabilities p. Intensity values are forest firesd per ten thousand km.

43.1.5. Second-order summary statistics

```
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)
```

Figura 43.5: K-functions for Nepal and Medellin

43.2. Spatial point patterns on linear networks

Over the last ten years point patterns on linear networks have received a lot of scientific attention. The initial explanation behind the consideration of linear networks, as state spaces of some point processes, might refer to the fact that objects that live on such structures can not use all space, and their movements strongly depend on their freedom over such structures (Okabe and Sugihara, 2012). Consequently, among other things, the spatial distribution of points as well as the correlation among points should be studied with respect to the underlying network. However, it has not been that easy to deal with this change of support when aiming to adapt statistical methodologies for the analysis of point patterns on linear networks. The main challenges were not only mathematical/statistical, but also computational (Moradi, 2018; Baddeley et al., 2021).

A linear network is a union of line segments $l_i = [u_i, v_i] = \{tu_i + (1-t)v_i : 0 \leq t \leq 1\} \subset \mathbb{R}^2$, and a common choice of metric over such structure has initially been the shortest-path distance $d_L(u, v)$, although later Rakshit et al. (2017) proposed other types of distances including Euclidean distance. Let Y be a point process on a linear network L, the Campbell formula (43.1) is modified as

$$\mathbb{E}\left[\sum_{y\in Y} f(y)\right] = \int_L f(z)\lambda(z)\mathrm{d}_1 z,$$

where d_1 denotes the integration with respect to arc length. Here, $\lambda(z)$ gives the expected number of points per unit length of L in the vicinity of $z \in L$. Several different kernel- and Voronoi-based approaches are developed to estimate the intensity function of point processes on linear networks by considering suitable metrics and handling mathematical challenges. We refer interested readers to Moradi (2018) and Baddeley et al. (2021) for more details, and briefly go through a fast non-parametric kernel-based intensity estimator using two-dimensional convolution (Rakshit et al., 2019). Given a realisation $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ from a point process Y on linear network L, they proposed

$$\widehat{\lambda}^{\mathrm{U}}_{\sigma}(z) = \frac{1}{c_{\sigma,L}(z)} \sum_{i=1}^{n} \kappa_\sigma(z - y_i), \qquad z \in L, \tag{43.8}$$

with a uniform correction, and

$$\widehat{\lambda}^{\mathrm{JD}}_{\sigma}(z) = \sum_{i=1}^{n} \frac{\kappa_\sigma(z - y_i)}{c_{\sigma,L}(y_i)}, \qquad z \in L, \tag{43.9}$$

with Jones-Diggle correction, where κ_σ is a bivariate kernel function with bandwidth σ, and

$$c_{\sigma,L}(z) = \int_L \kappa_\sigma(z - v)\mathrm{d}_1 v,$$

is an edge correction. The two estimators above provide similar statistical properties as their counterparts for spatial point patterns in $\mathbb{R}^2$ (i.e., estimators (43.1) and (43.2)), and they can be calculated quickly even on large networks and for large bandwidths. The quick computation is achieved by the fast Fourier transform (FFT) (Silverman, 1982). Moreover, Rakshit et al. (2019) proposed to use the adapted versions of Scott's rule and likelihood cross validation, which are accessible through the functions bw.scott.iso and bw.lppl from spatstat.core and spatstat.linnet, to obtain an optimum bandwidth. We add that the fast calculation of the above estimators further simplifies the calculation of adaptive kernel-based intensity estimators and relative risk over network structures (Rakshit et al., 2019).

We also recall that Moradi et al. (2019) proposed their voronoi-based sub-sampling approach for general point processes, and for point patterns on linear networks it can be computed by the function densityVoronoi.lpp from spatstat.linnet.

As a practical example for this section, we study the spatial distribution of street crimes in Valencia, Spain. Valencia is the third-largest city in Spain, being the capital of the autonomous community of Valencia . The urban territory of Valencia encloses an area of 134.65 km^2 with over 800000 inhabitants in the municipality. The dataset consists of the locations of 90247 street crimes such as assault (55610 cases), theft (25342 cases), theft against woman with violence (454 cases), and other types (8841 cases). These crimes are committed between 2010 to 2020. However, in what follows, we only focus on the data corresponding to the year 2020 as a case

study which only includes 6868 cases from which 4077 concerns assault, 2060 are theft, and 66 cases relate to crimes against women with violence. This dataset is owned by Generalitat Valenciana (GV) through the emergency phone 112, and was made available to the authors thanks to an agreement between GV and University Jaume I.

Next we turn to estimate the bandwidth using the counterparts of the Scott's rule of thumb whichs gives 584.1m. The function densityQuick.lpp from spatstat.linnet is used to compute either of the estimators (43.8) and (43.9) in which its default currently computes the uniformly-edge-corrected estimators (43.8).

```
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
```

The obtained intensity images in this case are of class linim, which we need to convert them to objects of class im prior to turn them into raster objects.

```
par(mfrow = c(1, 2))
plot(w_vlc$geometry, lwd = 4)
plot(valencia, pch = 20, main = "", lwd = 4, cex = 1, add = T, cols = "red",
↪  col = "blue")
plot(raster(as.im(d_scott_valencia)), main = "", axis.args = list(cex.axis =
↪  4), legend.width = 2, zlim = c(0, 6))
plot(w_vlc$geometry, add = TRUE, lwd = 4)
par(mfrow = c(1, 1))
```

Below, in Figure 43.6, we represent the estimated intensity together with the data points. Generally, the estimated intensity indicates the central and north-central parts of Valencia as high-risk areas jointly with some lower-risk hotspots in the east and coast of Valencia.

```
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)
```

```
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]])
```

Figura 43.6: Kernel-based intensity estimation (right), using the uniformly-edge-corrected estimators, for the crime data in Valencia, Spain, during 2020 (left). Intensity values are street crimes per km.

```
  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), xlab = "r", 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 43.7: K-function fot crime in valencia.

Parte X

Cómunica y colabora

Capítulo 44

Informes reproducibles con R Markdown y Quarto

Emilio L. Cano

44.1. Introducción

44.1.1. ¿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 figura 4.1. Llamemos genéricamente "informes" 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 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-pega*, 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 siempre bajo este paradigma de incluir el resultado a posteriori (Leisch, 2002).

Este enfoque provoca inconsistencias (por ejemplo entre unos grupos y otros, entre diferentes analistas, etc.), errores, contenidos desactualizados o no reproducibles, 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, entender mejor y verificar.

Un concepto muy relacionado que utilizamos en **R** es la "programación literaria"[2], mediante la cual combinamos un lenguaje de programación como **R** con documentación de todo tipo (por ejemplo comentarios en el código).

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 flujo de trabajo sería el siguiente: los contenidos se encuentran en ficheros de texto plano, con código y texto explicativo. Estos ficheros fuente, se compilan y producen los materiales en los formatos necesarios. Los cambios se hacen una vez, y todos los materiales son actualizados adecuadamente.

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 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 2011[3]. 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).

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.

44.1.2. Markdown, R Markdown, Quarto y RStudio

Markdown es un tipo de ficheros de texto pensado para que se pueda leer bajo cualquier circunstancia, con una sintaxis muy simple que permite leerlo directamente por las personas, o ser

[1] The goal of reproducible research is to tie specific instructions to data analysis and experimental data so that reports can be recreated, better understood and verified (Kuhn, 2019).

[2] Literate programming is a methodology that combines a programming language with a documentation language (Knuth, 1984).

[3] http://www.nytimes.com/2011/07/08/health/research/08genes.html

convertido por un ordenador en otro formato más elaborado, como por ejemplo HTML (página web) o Microsoft Word. En RStudio, se pueden crear ficheros **R Markdown**[4] 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[5], 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 necesitamos también el paquete `quarto` (Allaire, 2022) y tener instalado el software **quarto** en el ordenador[6].

Para crear un nuevo documento Quarto, se selecciona *Quarto Document...* en el icono 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. Podemos crear un archivo Quarto vacío si no queremos que nos cree 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 se genera 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) con una estructura básica a modo de plantilla. Los elementos principales de un archivo R Markdown aparecen en esta plantilla:

- Encabezado YAML: Constituyen la configuración del documento, y controlan sobre todo las opciones de salida, cómo será el resultado. 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.

- Texto formateado: Con una sintaxis muy sencilla, se puede dar formato al texto, como

[4] https://rmarkdown.rstudio.com
[5] https://quarto.org
[6] La instalación es trivial para cualquier sistema desde la web de quarto, https://quarto.org.

negritas, listas, etc. En el editor visual se puede hacer con los menús y botones de la barra de herramientas del editor.

- Fragmentos 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.

La barra de herramientas del editor ofrece algunas opciones:

- El botón *Render* "renderiza" 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 podemos cambiar el encabezado para poder renderizar el documento a distintos formatos. Por ejemplo si cambiamos el encabezado que se ha creado por defecto por el siguiente, podremos 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.
- Con las flechas de navegación podemos movernos entre los *chunks* del documento. También podemos 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.
- Podemos cambiar entre el editor vidual y el del código fuente con los botones *Source* y *Visual* en la parte superior del editor.

Para generar el documento, guardamos el archivo en cualquier carpeta de nuestro proyecto y utilizamos el icono de "renderizado". La figura **??** 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 hacemos clic en el triángulo de la derecha, podemos seleccionar el formato de salida (html, pdf o Word) si hemos incluido esos formatos en el encabezado YAML como se ha indicado. El formato PDF requieren tener instalada una distribución del sistema de edición libre LaTeX[7]. 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.

```
knitr::include_graphics("img/cdr_quarto.png")
```

Figura 44.1: Informe Quarto y su renderizado con algunas opciones adicionales

Podemos compilar el informe cuantas veces queramos con el icono *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 encontraremos todas las opciones disponibles (que sirven para los documentos Quarto). En los siguientes apartados repasaremos las opciones más habituales que cubren un amplio abanico de proyectos.

[7]Se puede instalar una distribución ligera de LaTeX con el paquete `tinytex`, ejecutando `tinytex::install_tinytex()` habiendo instalado dicho paquete.

44.2. Documentos Quarto

44.2.1. Encabezado YAML y configuración

Las opciones de configuración del documento se establecen en este encabezamiento. Al crear el documento con la plantilla, se nos ha creado el siguiente encabezado:

```
---
title: "Título del informe"
format: html
editor: visual
---
```

Ya hemos 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. El siguiente encabezado YAML fijaría el ancho y el alto de las figuras (en pulgadas) para el formato de salida html:

```
---
title: "Título del informe"
format:
  html:
    fig-width: 8
    fig-height: 6
  docx: default
---
```

Además del título (title), se pueden incluir autor (`author`) y fecha (`date`). Estos elementos son cadenas de texto que aparecerán al principio del documento de salida. Debemos 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 podemos indicar una plantilla personalizada para usar colores corporativos u otras opciones de diseño del informe. Para documentos html podemos 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/.

44.2.2. Formateado de texto

Incluir títulos, énfasis en el texto y listas es muy sencillo, y a menudo no se necesita nada más para realizar un informe. En el sencillo informe que se crea con la plantilla ya vemos algunas opciones:

- Los encabezados se crean poniendo al principio de la línea tantos símbolos almohadilla (`##`) como nivel de título queramos (dos almohadilla, 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 rodea con dos asteriscos a cada lado: `**negrita**`.
- Para poner texto en formato monoespaciado, tipo código, se rodea de acentos graves (*backticks*).
- Los enlaces se crean automáticamente si hay una url completa.

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); listas poniendo al principio de la línea un asterisco, guión o signo más; listas ordenadas poniendo un número y un punto al principio de la línea; enlaces con texto + link como: `[texto](url)`; imágenes de cualquier tipo como: `![](ruta_a_la_imagen)`; superíndices y subíndices; ecuaciones en formato `LaTeX`; tablas; saltos de línea; saltos de página. 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.

44.2.3. 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 acento grave (*backticks*, `) seguidos de unas llaves con la letra r dentro. El marcador de cierre del *chunk* son de nuevo tres acentos graves, sin más. Y dentro del chunk podemos poner expresiones de R como si estuviéramos en un script. Al 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 utilizado en otro capítulo del libro para incluir gráficos en el informe.

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

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, dentro de dos acentos 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 `
```

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 icono *Insert a new code chunk* de la barra de herramientas del editor. Una vez se tiene el cursor dentro de un chunk, podemos ejecutar una expresión como si estuviéramos en un script (`CTRL+ENTER`), o el chunk completo (`MAYUS+CTRL+ENTER`).

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 renderiza 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 botóh *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.

44.2.4. Opciones de los bloques de código (*chunks*)

Al renderizar un informe que contiene chunks sin configurar ninguna opción, por defecto mostrará en el informe 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 chunk mediante líneas que comienzan por el llamadao *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, fig.align='center' ", lo que indica que el código no se mostrará en el informe final y que el gráfico producido se alineará 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 documento, 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'
---
```

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 renderización; `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 codigo; `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. Para ver la lista completa de opciones, accede a la *R Markdown reference guide* que está disponible en el menú *Help/Cheatsheets*.

Una característica muy cómoda es usar la ayuda contextual al escribir las opciones del chunk. Si empezamos a escribir, o pulsamos `CTRL+ESPACIO`, se muestran las opciones disponibles, y al seleccionar una opción, si tiene varios posibles valores aparecen también para seleccionar.

44.2.5. 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`, 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 utilizamos las **referencias cruzadas** de los informes Quarto. Para poder referenciar un gráfico creado en un chunk, necesitamos: 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 excribiendo `@etiqueta`. Por ejemplo, el siguiente chunk crearía el gráfico de la figura 44.2, y en el texto lo referenciaríamos como "`Figura @fig-tic`".

Figura 44.2: Ventas vs. % Empresas con banda ancha

```{r}
#| label: "fig-tic"
#| fig-cap: "Ventas vs. % Empresas con banda ancha"
TIC2021 |>
  ggplot(aes(ebroad, esales)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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 crea la tabla. A modo de ejemplo, el siguiente chunk crearía la tabla 44.2 ya formateada con la función `flextable()` del paquete homónimo (Gohel and Skintzos, 2022), y en el texto la referenciaríamos como "`Tabla @tab-tic`".

GEMA DICE: ERROR EN LA TABLA POR LAS FUNCIONES, VER CON LA ACTUALIZACIÓN CDR

```{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()
```

44.3. 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 también podemos seguir creando con RStudio. En R base se pueden crear archivos `Sweave`, con sintaxis LaTeXpara 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 Word).

En cuanto a los formatos de salida, hay una cantidad de opciones muy interesante que queda fuera de los objetivos de este libro. Se relacionan a continuación algunos de ellos, se puede consultar el libro de Xie et al. (2019) y en 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 R Markdown además ioslides, slidify o 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*)
- **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 que sigamos 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 globa, 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.
- 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 45

Aplicaciones webs interactivas con Shiny

Aurora González Vidal

45.1. 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. Éstas son algunas de las razones principales por las cuales `Shiny` se ha vuelto tan popular a lo largo de los años, ya que con poco esfuerzo se pueden crear pruebas de concepto de un producto, mostrar algoritmos o presentar resultados de investigación de forma elegante a través de interfaces de usuario accesibles, reproducibles y amigables (Fay et al., 2021).

El primer paso para disfrutar de `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. use `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.
```

45.2. Partes mínimas 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.

Estas partes pueden encontrarse en el mismo script o estar separadas en dos scripts con nombres fijos: `ui.R` y `server.R`. En este caso se ha elegeido la segunda opción para ilustrar los ejemplos con mayor claridad. Una aplicacion `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(
  headerPanel("TITULO"),
  sidebarPanel(),
  mainPanel()
))
```

- `server.R`

```
shinyServer(function(input, output) {})
```

Para lanzar la aplicación existen dos maneras, 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")
```

O 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`, construiremos una aplicación para la visualización de algunos gráficos presentados en el capítulo "Perfilado estadístico de parados" {#paro-clm}.

Además del `ui.R` y el `sever.R`, puede ser muy útil tener un fichero `source.R` en la misma carpeta para cargar los paquetes y datos estáticos necesarios para el funcionamiento de la aplicación.

45.3. 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 este apartado veremos dos muy sencillos:

- sidebarLayout(): para colocar un sidebarPanel() 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.

45.3.1. Diseño de las páginas: fluidPage()

Un diseño de página fluido `fluidPage()` consiste en filas que a su vez incluyen columnas. Las filas existen con el propósito de asegurar que sus elementos aparezcan en la misma línea (si el navegador tiene el ancho adecuado). Las columnas existen con el propósito de 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()` presenta 2 argumentos: `headerPanel()` con el título de la aplicación, y `sidebarLayout()`.`sidebarLayout()` es un punto de partida útil para la mayoría de las aplicaciones. Este a su vez tiene 2 argumentos más: `sidebarPanel()`, que es una barra lateral para las entradas y `mainPanel()`, una gran área principal para la salida.

```
shinyUI(fluidPage(
  headerPanel("Evolución del paro"),
  sidebarLayout(
    sidebarPanel(
      radioButtons(
        "vble", "Variable",
        c(
```

```
          "sexo" = "sexo",
          "tramo_edad" = "tramo_edad",
          "tiempo_busqueda_empleo_agregado" =
          ↪  "tiempo_busqueda_empleo_agregado",
          "sector" = "sector",
          "tiempo_busqueda_empleo" = "tiempo_busqueda_empleo"
        ), "sexo"
      )
    ),
    mainPanel(
      plotOutput("gra1")
    )
  )
))
```

Figura 45.1: fluidPage, sidebarLayout, sidebarPanel por defecto

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'` y éste es el resultado.

```
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.

Figura 45.2: fluidPage, sidebarLayout a la derecha

45.3.2. Segmentación de diseños: tabsetPanel() y navlistPanel()

Para subdividir el panel principal en varias secciones discretas, se puede usar `tabsetPanel()` y 'tabPanel()" como sigue:

```
mainPanel(
  tabsetPanel(
    tabPanel("Plot1", plotOutput("gra1")),
    tabPanel("Plot2", plotOutput("gra2")),
    tabPanel("Plot3", plotOutput("gra3"))
  )
)
```

`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.

```
ui <- fluidPage(
  titlePanel("Application Title"),
  navlistPanel(
    "Header A",
    tabPanel("Component 1"),
    tabPanel("Component 2"),
    "Header B",
    tabPanel("Component 3")
  )
)
```

45.4. Elementos para entrada 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 crean con la función `sliderInput` que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
sliderInput(inputId, label, min, max, value, step = NULL, animate = FALSE)
```

Sus características incluyen:

```
- La posibilidad de introducir un único valor y rangos

- Formatos customizaods (por ejemplo para entradas relativas al dinero)

- Pueden ser animados y recorrer los valores de forma automática (argumento `animate`)
```

Algunos ejemplos son:

```
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 da una lista de opciones entre las cuales solo se puede seleccionar una. En `Shiny` se crean con la función `radioButtons` que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
radioButtons(inputId, label, choices, selected = NULL)
```

Un ejemplo donde la variable "sexo" está elegida por defecto se puede ver en el primer trozo de código de la subsección 2.1.1.1 sidebarLayout().

- **Selección múltiple**: Un cuadro de selección múltiple es un tipo de selector que da una lista de opciones entre las cuales se pueden seleccionar varias. En `Shiny` se crean con la función `selectInput` que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
selectInput(inputId, label, choices, multiple = FALSE)
```

```
selectInput("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 utilizars 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", "Numero 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 áreas de texto: `textAreaInput()`.

Figura 45.3: Distintos elementos para la entrada de datos

45.4.1. Lectura de ficheros de datos

También es posible introducir información a través de la lectura de ficheros de datos con la función `fileInput()`. Podemos 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 (seaprador, decimal, cabecera). En el siguiente ejemplo se utiliza la función `read.csv` y como separador siempre punto y coma, mientras que los demás elementos se dan a elegir.

```
shinyUI(fluidPage(
  headerPanel("Lectura de datos"),
  sidebarPanel(
    h4("Cargar fichero CSV"),
    fileInput("file1", "",
      accept = c("text/csv", "text/comma-separated-values,text/plain", ".csv")
    ),
    checkboxInput("header", "Header (el csv tiene nombres de variables)",
    ↪  TRUE),
    radioButtons(
      "dec", "Separador de decimales:",
      c(
        "Punto" = ",",
        "Coma" = "."
      )
    )
  ),
  mainPanel(
    tabsetPanel(
      tabPanel(
        "CSV",
        h4("Vista del fichero CSV"),
        tableOutput("contents")
      )
    )
  )
))
```

45.5. 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 esos objetos se utilizan las funciones `renderX` donde X es el tipo de objeto a devolver. Por úlitmo, este objeto se referencia nuevamente en el `ui.R` en el lugar que se desea mostrar (panel) a través de la expresión `XOutput`.

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 45.4: Lectura de datos

Server	Ui	Crea
renderImage	imageOutput	Imagen
renderPlot	plotOutput	Grafico
renderTable	tableOutput	Tabla
renderText	textOutput	Texto
	htmlOutput	HTML
	verbatimTextOutput	Texto verbatim

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 posibildiades son se pueden ver en la Tabla @ref(tab:my_table).

- **Gráficos**: Para generar la aplicación de la Figura 45.1, se utiliza `renderPlot` en el server.R como sigue:

```
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 el`source.R` que se carga al principio. Lo que interesa de esta función es que tiene como único argumento el nombre

de la variable por el cual crear el gráfico, que puede ser una de las diversas opciones que se dan a través del radioButton vble 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 Figura 45.4 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 =
    ↪  ";")
  })
})
```

45.6. Reactividad

El modelo de reactividad que utiliza `Shiny` es el siguiente: hay una fuente reactiva, un conductor reactivo y un punto final de la reactividad. La fuente reactiva suele ser lo que el usuario introduce y el punto de parada lo que se muestra por pantalla. A lo que el usuario introduce se accede con el objeto input y a lo que se muestra por pantalla con el objeto output. 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.

45.6.1. 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 contendrá, 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 = "")
      )
    })
  })
})
```

Figura 45.5: Barras

45.7. 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.

Se dará una introducción muy breve a cada uno de ellos y enlaces para que el lectro pueda indagar en profundidad

- Shinyapps.io

Rstudio ofrece un servicio de *hosting* 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.

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 utiliza `deployApp()` como sigue

```
library(rsconnect)
deployApp()
```

Para más información sobre este método, consulta 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). Si no se está utilizando una distribución con soporte explícito, se puede aún así utilizar construyéndolo desde el paqeute fuente.

En el mismo `Shiny` Server se pueden hospedar múltiples aplicaciones `Shiny`. Para ver instrucciones detalladas para su instalación y configuración, se recomienda la guía `Shiny` Server https://docs.rstudio.com/shiny-server.'

- 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

Para ello, la plataforma de publicación RStudio Connect puede ser una solución. Esta herramienta permite compartir aplicacciones `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.

45.8. 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 mencionarán algunos de ellos pero, sobre todo, se econmienda al lector els visitar dichos repositorios para una mayor profundidad en este tema.

- shinydashboard, shinydashboardPlus y flexdashboards

En temas de estilo, se destacan éstos 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* personalizados y diversos componentes para mejorar las aplicaciones. Se pueden reemplazar los checkboxes por switch buttons, añadir colores a los 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. Envolviendo una salida Shiny en withSpinner(), el *spinner* aparecerá automáticamente mientras la salida se recalcula. Hay 8 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), se muestran 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 entrada 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 repasa el modelo reactividad, es decir, cómo al cambiar algo en los parámetros de entrada de forma dinámica cambia la salida y cómo controlarlo y se muestran distintas opciones para la publicación de las aplicaciones `Shiny`. Por último, se mencionan algunas de las posibles extensiones al paquete.

Capítulo 46

Git y GitHub en R

Michal Kinel

46.1. ¿Qué es Git?

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para gestionar desde proyectos pequeños hasta muy grandes con rapidez y eficacia (Chacon, 2009).

Es esencial usar Git en la colaboración con otros desarrolladores en un proyecto de codificación o trabajar un proyecto propio.

Para comprobar si Git ya se encuentra instalado en el ordenador basta con escribir `git --version` en el terminal de RStudio.

Si Git ya está instalado en su equipo, la consola proporcionará su versión. Si no se encuentra instalado, se puede obtener en el sitio oficial de Git y seguir fácilmente las instrucciones de descarga para instalar la versión más adecuada para el sistema operativo.

Comandos de Git

- `git init` - Crea un repositorio Git vacío en el directorio especificado. Ejecutar sin argumentos para iniciar el directorio actual como un repositorio Git.
- `git clone <repo>` - Clona el repo ubicado en en la máquina local. El repositorio original puede estar ubicado en el sistema de archivos local o en una máquina remota a través de HTTP o SSH.
- `git config user.name <nombre>` - Define el nombre del autor que se utilizará para todas las confirmaciones en el repositorio actual. Los desarrolladores suelen utilizar `--global` para establecer las opciones de configuración para el usuario actual.
- `git add <directorio>` - prepara todos los cambios en `<directorio>` para la siguiente confirmación `<commit>` de la versión del código. Sustituya `<directorio>` por un `<fichero>` para cambiar un archivo específico.
- `git commit -m "<mensaje>"` - Confirma la instantánea preparada, pero en lugar de lanzar un editor de texto, utilice `<mensaje>` como mensaje de la confirmación `<commit>`.
- `git status` - Enumera los archivos que están preparados `<staged>`, no preparados `<unstaged>` y sin seguimiento `<untracked>`.
- `git log` - Muestra todo el historial de confirmaciones `<commits>` utilizando el formato por defecto. Para la personalización vea las opciones adicionales.
- `git diff` - Muestra los cambios no preparados `<unstaged>` entre el índice y el directorio de trabajo.
- `git remote add <nombre> <url>` - Crea una nueva conexión a un repositorio remoto. Después de añadir un remoto, se puede usar como un atajo para `<url>` en otros comandos.
- `git fetch <remote> <branch>` - Obtiene un `<branch>` específico, desde el repositorio. Si se deja un `<branch>` para obtener todas las referencias remotas.
- `git pull <remote>` - Obtiene la copia remota especificada de la rama actual y la fusiona inmediatamente con la copia local.
- `git push <remote> <branch>` - Empuja la rama a `<remote>`, junto con los commits y objetos necesarios. Crea la rama nombrada en el repo remoto si no existe.

46.2. ¿Qué es GitHub?

GitHub es un servicio de alojamiento de repositorios Git que ofrece una interfaz gráfica basada en la web. Es la comunidad más popular para que los desarrolladores compartan código y trabajen juntos en proyectos. Es gratuito, fácil de usar y se ha convertido en el centro del movimiento hacia el software de código abierto. Los programadores pueden encontrar códigos fuente en muchos lenguajes diferentes, entre ellos R, y utilizar la interfaz de línea de comandos, Git, para hacer y mantener un seguimiento de los cambios.

GitHub ayuda a todos los miembros del equipo a trabajar juntos en un proyecto desde cualquier lugar y facilita la colaboración (Astigarraga and Cruz-Alonso, 2022). También se pueden revisar

versiones anteriores creadas en un momento anterior.

46.3. ¿Por qué usar Git y GitHub?

La integración de Git y GitHub en el desarrollo de nuestros proyectos en R, sean grandes o pequeños. Gracias a Git, todo el proceso de desarrollo de código está versionado, con el histórico en en el ordenador (en local) y GitHub centraliza todos los cambios online (en remoto).

Las principales ventajas de utilizar Git y GitHub son:

1. Seguridad: proporciona una copia de seguridad para nuestro código y un control sobre los cambios que vamos introduciendo.
2. Contribución a tus proyectos: facilita la colaboración con los compañeros de trabajo o la comunidad de GitHub cuando utilizamos los repositorios públicos o privados, previa asignación de acceso. Además, GitHub es una de las mayores comunidades de codificación que existen ahora mismo, por lo que es una amplia exposición para tu proyecto.
3. Documentación: GitHub posee una excelente documentación. Su sección de ayuda y guías pueden resolver casi cualquier duda.
4. Markdown: permite utilizar un sencillo editor de texto para escribir documentos con formato, de forma que se puede enriquecer los proyectos de un texto explicativo para que los demás usuarios o el propio desarrollador (pasado un tiempo) pueda identificar rápidamente las características del proyecto.
5. Seguimiento de los cambios en tu código a través de las versiones: cuando varios desarrolladores colaboran en un proyecto, es difícil hacer un seguimiento de las versiones: quién ha cambiado qué, cuándo y dónde se almacenan estos archivos. GitHub se encarga de este problema manteniendo un registro de todos los cambios que se han enviado al repositorio.
6. Integración: al tratarse de una plataforma tan ampliamente utilizada, GitHub puede integrarse fácilmente con plataformas como Amazon o Google Cloud o resaltar la sintaxis en más de 200 lenguajes de programación diferentes.

46.4. Configuración

En este apartado va a explicar de forma sintetizada cómo configurar un repositorio de Git en RStudio y vincularlo a un proyecto en GitHub. Antes de comenzar, se va a asumir el proyecto de RStudio se denomina `mi_proyecto.Rproj` y que el usuario de GitHub es `\@mi_usuario`.

Git utiliza una terminal llamada Git Bash que en su esencia es un conjunto de programas de utilidad de línea de comandos que están diseñados para ejecutarse en un entorno de línea de comandos estilo Unix. En el caso de que el equipo no tenga instalado Git, está disponible en la página oficial de Git, donde se encuentra el software y el manual de ayuda. RStudio

permite configurar Git Bash como la terminal por defecto, basta con acceder a Tools -> Global options... -> Terminal -> New terminals open with. La forma más fácil de utilizar la línea de comandos en RStudio es utilizar la pestaña Terminal justo al lado de la pestaña de la consola en el panel inferior izquierdo.

Figura 46.1: Terminal de RStudio

NOTA
Una cosa a tener en cuenta es que GitHub tiene un límite de tamaño de archivo único de 100mb. Por ello será importante configurar bien el archivo .gitignore y utilizar ficheros comprimidos o compartir los datos por medio de alguno de los servicios almacenamiento de datos en la nube.

46.5. Configurar git

Git asocia el nombre y dirección de correo electrónico con cada **commit**, lo que ayuda cuando varias personas colaboran en un proyecto. Para configurar tu nombre y dirección de correo electrónico en git, abra el Terminal y ejecute:

```
git config --global user.name 'Tu Nombre'
git config --global user.email tu_correo@email.com'
```

En un macOS, es aconsejable configurar Git para que recuerde la del usuario:

```
git config --global credential.helper osxkeychain
```

El siguiente paso es generar la llave SSH para el equipo en el que se va a desarrollar el trabajo y añadirla en la cuenta de GitHub. Primero se accede a Tools -> Global Options... -> Git/SVN y pulsamos Create SSH key, se selecciona SSH key type como ED25519. Como función opcional se puede incluir una contraseña aunque luego resulta incordioso al hacer commits, por lo que si la llave se configura en un ordenador personal no hace falta. Se copia la dirección donde se crea la llave, que por defecto se crea en la carpeta .ssh de usuario, (por ejemplo en Windows: C:/Users/mi_user/.ssh/id_ed25519) y se pulsa en create. Tras realizar estos pasos se introduce en la terminal:

```
clip < C:/Users/mi_user/.ssh/id_ed25519.pub
# Copia el contenido del archivo id_ed25519.pub en el portapapeles
# ruta con instalación de R y RStudio con todas las opciones por defecto:
↪  ~/.ssh/id_ed25519.pub
```

Una vez copiada la llave SSH se accede a GitHub dando clic en el perfil de usuario a Settings luego en el menú de Access, posteriormente en SSH and GPG keys y se selecciona la opción pulsando New SSh key y aparece el menú de creación de llave SSH. En Title se introduce un nombre identificativo como por ejemplo, Mi pc personal, en Key type se deja la opción por defecto: Authentication Key, y en el recuadro de Key se pega la llave SSH copiada con la función clip del terminal. la llave se tiene que parecer al ejemplo inferior:

```
ssh-ed25519
↪  A44AC3NzaC1lZDI1NTE5UGHJIGVuY9LWIcUX+CbtPKsO6yERjp5fFbCaA8QPM1qbLsGU
↪  mi_user@NOMBRE_DE_EQUIPO
```

Para terminar la configuración de la llave se confirma dando clic en Add SSH key. Tras realizar este proceso la autorización del equipo queda finalizada y el equipo se encuentra listo para usar Git en GitHub.

Ahora se procede a convertir el proyecto ejemplo, mi_proyecto.Rproj, en un repositorio Git, para ello se abre el proyecto proyecto y simplemente se ejecuta en la terminal el siguiente comando:

```
git init
```

Esto creará un repositorio git en el directorio de trabajo. Debido a que se está utilizando la línea de comandos, RStudio normalmente no reconocerá inmediatamente el cambio, por lo que se debe cerrar RStudio y volver a abrir el programa. Para reabrir el proyecto, se puede hacer doble clic en el archivo mi_proyecto.Rproj o utilizar el menú desplegable para la gestión de proyectos en la esquina superior derecha del propio RStudio. Una vez reabierto el proyecto, aparecerá una pestaña adicional "Git" situada en el panel superior derecho.

Figura 46.2: Pestaña de Git de RStudio

La pestaña de Git mostrará los ficheros creados y que aún no han sido versionados por git, así como los archivos que han sido modificados o eliminados desde el último commit. El icono del signo de interrogación para el estado de cada archivo me muestra que ninguno de los archivos de mi directorio está siendo versionado actualmente. Se seleccionan los ficheros para el versionado y se pulsa el botón de commit. Se abrirá una ventana con los detalles, se introduce el mensaje del commit que resume el cambio registrado. Como se trata del primer commit del proyecto, el mensaje podría ser el siguiente: Primer commit del proyecto y pulsamos commit.

Figura 46.3: Ventana de confirmación de RStudio

Una vez obtenido el seguimiento de los cambios en el equipo de trabajo local, el siguiente paso es hacer un repositorio en el perfil de GitHub.

Para configurar un nuevo repositorio en GitHub, basta con dirigirse al menú desplegable "+" en la parte superior derecha de tu página de perfil de GitHub y seleccionar la opción New repository. El cuadro de diálogo del nuevo repositorio tiene varias opciones, como requisitos mínimos se introduce un nombre para el repositorio, por ejemplo **mi_proyecto.** En esta etapa se puede seleccionar si el proyecto es público o privado, no obstante, ésta opción se puede modificar posteriormente en las opciones del repositorio.

Una vez creado el repositorio, GitHub proporcionará unas instrucciones de cómo comenzar a usarlo. En caso de comenzar con los datos locales las instrucciones se encuentran en **...or push an existing repository from the command line**, siguiendo con el ejemplo:

```
git remote add origin git@github.com:mi_usuario/mi_proyecto.git
git branch -M main
git push -u origin main
```

Se copia el código proporcionado por GitHub en la terminal de RStudio del proyecto abierto y se ejecuta. Estos comandos de Git configurarán GitHub como el repositorio remoto origin y empujará, push, todos los commits locales existentes a él. A continuación, se puede actualizar la página del repositorio de GitHub y ver que todos los archivos y el código del proyecto se encuentran en el repositorio. Como los comandos fueron ejecutados por la terminal hay que reiniciar RStudio, para que al reiniciar aparecerán operativas las flechas con Pull y Push. Ahora

ya está disponible realizar la operativa del flujo de trabajo básico de git de pull-commit-push usando la pestaña git de RStudio.

46.6. Workflow

El flujo de trabajo típico es el siguiente - se crea/edita/modifica un archivo dentro del repositorio (Jenny Bryan, 2021). Se hace el commit de los cambios, lo que crea una instantánea permanente del archivo en el directorio Git junto con un mensaje del commit para indicar qué cambio se realiza en el código.

Cuando se empieza un nuevo proyecto, los archivos del directorio de trabajo no tienen seguimiento, primero es necesario añadirlos al repositorio antes de que Git pueda hacer un seguimiento de ellos y de su historia.

Una vez creado el commit con su mensaje correspondiente se debe de hacer push desde el asistente de RStudio o en la terminal con git push.

Cuando se trabaja con otros usuarios o se utilizan varios ordenadores para trabajar con un mismo proyecto es importante comenzar el trabajo ejecutando el comando git pullen la terminal para descargar todos los commits del remoto.

El mensaje del commit es muy importante, porque en el futuro ayudará a identificar los cambios realizados por el desarrollador o un groupon de ellos. Para facilitar la lectura y el seguimiento de los commits se debe de seguir ciertos consejos establecidos por la comunidad:

- Utilizar el modo imperativo en la línea del asunto, al igual que en una receta de cocina el cambio tiene un fin particular, por ejemplo: *Fix bug in my_function o Corregir una errata en la descripción del proyecto*,
- El asunto debe empezar por mayúscula, tener aproximadamente como máximo 50 caracteres y no terminar en un punto
- Si es necesaria una descripción más detallada, se incluye tras una línea en blanco bajo el encabezado. Cuando se escribe el cuerpo de un mensaje de confirmación, se debe de tener en cuenta su margen derecho, y envolver el texto manualmente. La recomendación es hacerlo a los 72 caracteres, para que Git tenga suficiente espacio para sangrar el texto, pero manteniendo todo por debajo de los 80 caracteres. Además se debe de utilizar la descripción para responder preferentemente a qué y el por qué frente al cómo.

Capítulo 47

Geoprocesamiento en nube

Dominic Royé

Habitualmente se obtiene los datos desde cualquier proveedor lo que implica la descarga de grandes volúmenes. No obstante, en la actualidad existen servicios de geoprocesamiento en nube que nos ayudan a hacer análisis online sin necesidad de descargar los datos ni preocuparnos por el rendimiento del cálculo. 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 con capacidades de análisis a escala planetaria.

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 nos hace puente entre R y la API.

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 proporciona la 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 rellenadas por interpolación (https://developers.google.com/earth-engine/datasets/catalog/NOAA_CDR_OISST_V2_1) (Reynolds et al. (2008)).

El objetivo de este caso práctico es mostrar el potential del uso de APIs directamente en R. El resultado se empleará en el capítulo sobre datos del cambio climático como visualización avanzada.

47.1. 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 nuestro 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()`).

- *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 que se aplican son similares a lo que se conoce 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.

47.2. Primeros pasos

```
require("rgee", character.only = TRUE) || install.packages("rgee", repos =
↪  "https://cloud.r-project.org")

library(rgee)
```

Después de darse de alta en GEE y haber instalado *CLI gcloud* en el sistema operativo, sea crea un entorno virtual de Python con todas las dependencias de GEE usando la función `ee_install()`.

```
ee_install() # crear entorno virtual de Python
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() # autenticar e inicializar GEE
ee_user_info() # inf sobre usuario
```

Hay que tener en cuenta que únicamente cuando se envian tareas GEE ejecuta el cálculo en los servidores enviando todos los objetos creados. En la mayoría de los pasos se crean objetos

EarthEngine, los que se usan una vez que construimos un mapa interactivo, la exportación o impresión en consola de un objeto.

Por ejemplo, se puede selecionar la banda NDVI del producto MODIS MOD13A2 e imprimir los metadatos de 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())
```

47.3. Cálculo de anomalias

47.3.1. Definiciones previas

Los datos NOAA CDR OISST contienen la temperatura superficial de los océanos a nivel global, por eso, se fija un rectángulo con sus cuatro puntos cubriendo la extensión del Mar Mediterráneo.

```
# 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 próximo 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("2022-08-31") # fecha final
```

Se puede acceder a todas las colecciones (*ImageCollection*) indicando su identifcación. Además, filtramos y recortamos los datos con respecto al periodo y a la extensión fijada. Finalmente se selecciona la banda o variable de de interés 'sst' (*surface sea temperature*).

```
collection_era5 <- ee$ImageCollection("NOAA/CDR/OISST/V2_1")$
  filterDate(startDate, endDate)$
  filterBounds(geom)$
  select("sst")
```

Procedemos a calcular el número de años en el período fijado.

```
numberOfyears <- endDate$difference(startDate, "years")$round()
```

47.3.2. Promedio estival

Después de las definiciones necesarias se crea una nueva colección con el promedio estival de cada año en el período. Para ello se hae una lista de los años sobre la que se mapea otra función. Esta función se debe pasar con `ee_utilspyfunc()`, la 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 crea nuevas colecciones es importante fijar la nueva fecha lo que se hace en `set()`.

```
yearly <- ee$ImageCollection(
  ee$List$sequence(0, numberOfyears$subtract(1L))$
    map(ee_utils_pyfunc(function(dayOffset) {
    start <- startDate$advance(dayOffset, "years")
    end <- start$advance(1L, "years")
    return(collection_era5$
      filterDate(start, end)$
      filter(ee$Filter$calendarRange(6L, 8L, "month"))$
      mean()$
      multiply(0.01)$
      set("system:time_start", start$millis()))
  }))
)
```

En el próximo 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(6L, 8L, "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.

```
anom <- yearly$map(ee_utils_pyfunc(function(im) {
  return(im$subtract(msst)$set(
    "system:time_start",
    im$get("system:time_start")
  ))
}))
```

Es posible crear un mapa interactivo de un año concreto aplicando la función `Map.addLayer()`. 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 que debe realizar GEE. Lo que falta por hacer es exportar las anomalías de cada año en formato geotiff. En el bucle *for* se selecciona cada año y se reduce una única capa. La función `ee_image_to_drive()` exporta los datos a *Google Drive*. El argumento *escale* indica con qué resolución se exporta. Aunque la resolución de los datos originales es de 27 km, se fija una de 2 km lo que implica una interpolación en la exportación por razones de estética en la visualización.

Figura 47.1: Mapa interactivo del mar Mediterraneo (1982)

```
for (i in 1982:2022) {
  select_data <- anom$filter(ee$Filter$calendarRange(i, i, "year"))$
    reduce(ee$Reducer$mean())

  text <- paste("ssd_anom_yr", i)

  task_img <- ee_image_to_drive(
    image = select_data,
    description = text,
    fileFormat = "GEO_TIFF",
    folder = "sst_med",
    region = geom,
    scale = 2000,
    fileNamePrefix = text,
    maxPixels = 10e9
  )

  task_img$start()
}
```

Las siguientes funciones permiten conocer el estado de las tareas.

```
ee_monitoring(eeTaskList = TRUE)
ee_check_task_status(quiet = FALSE)
```

Parte XI

Casos de estudio en ciencia de datos

Capítulo 48

Análisis de una red criminal

F. Liberatore, L. Quijano-Sánchez*, M. Camacho-Collados*

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*. Para ello se usa la librería `igraph`.

48.1. El *dataset 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 caso Italiano contra un grupo mafioso. La investigación duró del 2000 al 2006, y se enfocó en mas de 50 sospechosos involucrados en trafico 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 con 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 consideran 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.

48.2. Creación del grafo

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, 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 eaeb052 UN-- 182 247 --
#> + attr: name (v/c)
#> + edges from eaeb052 (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*).

48.3. Visualización del grafo

Para hacernos una idea de que aspecto tiene nuestro grafo, se procede a realizar una visualización, usando el comando *plot* de R.

```
plot(net, asp = 0)
```

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, excluimos los nombres de la visualización.

```
plot(net, vertex.size = 2, vertex.label = c(""), asp = 0)
```

Se puede ver como el grafo superior 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.

48.4. Importancia de los actores

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 incidentes al nodo. Cuanto más alto 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 <- igraph::degree(net) # Centralidad de grado
btwn <- igraph::betweenness(net) # Centralidad de intermediacion
```

A continuación, se muestran los actores con los valores más altos en cada medidad 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
```

Se pueden usar las medidas de centralidad 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.

```
vertex_filter <- dgr > 1 # deteccion actores marginales
scaled_btwn <- 0.1 + 4.9 * btwn / max(btwn) # Escalado del tamaño del nodo en
↪  funcion de la intermediacion
net2 <- igraph::induced.subgraph(net, which(vertex_filter)) # creacion
↪  subgrafo
plot(net2, vertex.size = scaled_btwn[vertex_filter], vertex.label = c(""),
↪  rescale = T, asp = 0) # visualizacion subgrafo
```

Gracias a la medida de centralidad se puede tener una mejor idea de cómo se configura la red respecto a sus actores más importantes.

48.5. Identificación de comunidades

Se procede a identificar las comunidades existentes en el grafo de la operación *Overdrive*.

igraph proporciona un 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 <- igraph::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
```

Al algoritmo identifica distintas comunidades, cada una con su número asignado.

48.6. Visualización de comunidades

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 funcion de
↪  su comunidad
V(net2)$label <- V(net2)$name
V(net2)$label.cex <- (1 + scaled_btwn[vertex_filter]) / 6 # Escalado del
↪  nombre en funcion de su centralidad
V(net2)$label.color <- "black"

# Definicion del color de las aristas en funcion 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 ejecucion
```

Se puede mejorar aun 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*.

```
l1 <- igraph::layout_with_fr(net2) # algoritmo Fruchterman-Reingold
plot(net2, layout = l1, asp = 0, edge.curved = 0.5) # Los resultados puede ser
↪  distintos con cada ejecucion
```

Finalmente, se puede exportar el grafo como PDF usando el comando *pdf* de R.

```
pdf("grafo_final.pdf")
plot(net2, layout = l1, asp = 0, edge.curved = 0.5) # Los resultados puede ser
↪  distintos con cada ejecucion
dev.off()
#> agg_png
#>       2
```

Capítulo 49

Optimización de inversiones publicitarias

Carlos Real Ugena

49.1. 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. Está medición será 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 tiene cada uno de ellos. Estas metodologías se pueden clasificar en cuatro grandes grupos como muestra una investigación llevada a cabo por Gartner:

1. **Marketing Mix Modeling (MMM)**: son modelos de series temporales que sirven para estimar la contribución del marketing y otros drivers en ventas desde un punto de vista estratégico.
2. **Multitouch Attribution (MTA)**: son modelos de atribución que valoran cada punto de contacto (Touchpoint) del recorrido del cliente (Customer Journey) asignando a cada uno un peso en la conversión. Son modelos tácticos que, normalmente, se centran en el canal online.
3. **Holdout Testing o Experiments(EXP)**: son modelos causales que evalúan el impacto de una campaña publicitaria a partir de una muestra de control y test.
4. **Unified Measurement Approaches (UMA)**: es la combinación de los modelos anteriores (MMM+MTA+EXP) con el objetivo de tener una visión unificada de los resultados.

En la comparativa (figura 14.1) que se muestra a continuación se resume el objetivo de cada uno de los modelos, así como las preguntas que permiten responder:

Figura 49.1: Comparativa de metodologías de medición

Por otra parte, 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.

La situación anterior está provocando que el primer tipo de enfoque, Marketing Mix Modeling (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 ha desarrollado o capacidades funcionales. Se detalla a continuación las principales características de cada una de ellas:

- **Robyn**: desarrollado por Meta, está pensado para datasets con gran cantidad de variables independientes dado que trabaja con regresiones ridge, las cuales están pensadas para lidiar

con problemas de multicolinealidad, muy presentes en este tipo de análisis. Cabe destacar que, entre las iteraciones 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**: 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 ad-stock, 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 poseen outputs más elaborados como el resto de soluciones. Por último, permite también realizar optimizaciones de presupuesto.

- **Orbit**: 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 misma 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 más frecuencia. No incluye la posibilidad de optimizar presupuestos.

49.2. Robyn como alternativa open-source en R

Robyn está ganando en popularidad debido a las mejoras continuas que están aplicando desde Meta, así como la adaptabilidad de la solución a la realidad del anunciante. Además, existe un paquete en R validado y subido al CRAN que lo sitúa como una gran alternativa open-source en R para iniciarse en el campo de la medición y optimización del retorno de las inversiones publicitarias. Se basa en regresiones ridge y la fórmula empleada es la siguiente:

$$y_n = \beta_0 + SCurve(x, j) + \beta_{hol} hol_t + \beta_{sea} sea_t + \beta_{trend} trend_t + ... + \beta_{ETC} ETC_t + \epsilon \quad (49.1)$$

donde:

$$
\begin{aligned}
\beta_0 &= \text{Intercepto} \\
SCurve(x,j) &= \beta_j \times \frac{x^{\alpha}_{decay_{t,j}}}{x^{\alpha}_{decay_{t,j}} + \gamma^{\alpha}} \text{(curva no lineal)} \\
X_{decay_{t,j}} &= X_{t,j} + \theta_j X_{decay_{t,j-1}} \text{(ad-stock)} \\
y_t &= \text{ventas en el instante } t \\
t &= \text{instante } t \text{ de las variables (semanas)} \\
j &= \text{indice del medio (por ejemplo,TV o Display)} \\
\beta, \alpha, \gamma, \theta &= \text{regresores de cada medio } j \\
\theta &: \text{implementada en la SCurve, donde } \theta_{tran} = cuantil(X_{decayj,\gamma}) \\
hol &= \text{festivos, } sea = \text{estacionalidad, } trend = \text{tendencia} \\
\beta_{ETC} ETC_t &= \text{resto variables independientes (precio, promociones, etc)} \\
\epsilon &= \text{termino de error}
\end{aligned}
\tag{49.2}
$$

A continuación, se aplicará Robyn sobre un ejemplo con información simulada del sector hotelero. En Github se cuenta con toda la información sobre cómo aplicar esta metodología.

Cabe destacar que en el ejemplo se ha utilizado la versión 3.6.3 que está disponible en el CRAN o se puede descargar desde Github. El objetivo del ejercicio será revisar 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.

Lo primero, cargue el paquete Robyn, chequee que está trabajando con la versión 3.6.4 y fuerce el uso de multicore al utilizar Rstudio:

```
library(Robyn)
packageVersion("Robyn")
Sys.setenv(R_FUTURE_FORK_ENABLE = "true")
options(future.fork.enable = TRUE)
```

A continuación, cargue una librería de Python llamada Nevergrad que utilizará en el proceso de estimación de hiperparámetros, invocándola desde R. Hay varias opciones para llevar a cabo esta instalación, una de ellas es instalando primero el paquete Reticulate e instalarla 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 el caso de que encuentre alguna dificultad, puede consultar en este enlace, ya que existen distintas alternativas para la instalación de Nevergrad.

Ahora cargue la tabla con la que medirá el impacto de la publicidad:

```
# hotel_tablonsemanal <- read_csv("data/212053_publicidad_dataset_hotel.csv")
library("CDR")
data("hotel_tablonsemanal")
head(hotel_tablonsemanal)
```

Se detalla a continuación lo 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.
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 nuestro caso consideramos el Black Friday.

En resumen, en el ejercicio estimará el impacto de cada una de las variables sobre las reservas, poniendo foco especial en las variables de medios: TV, Resto medios offline, Paid Search, Display y Online Video. Se quiere medir el efecto de cada una de ellas para posteriormente poder optimizar las inversiones.

Cargue los festivos de una tabla auxiliar:

```
library("CDR")
data("festivos")
head(festivos)
```

Ahora fije dónde se guardarán los resultados:

```
robyn_object <- "data/MyRobyn.RDS"
ruta_outputs <- "data"
```

Y trabaje en 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=dia de la semana" &
↪  "holiday=festivos"
  , prophet_signs = c("default", "default", "default"),
  prophet_country = "ES" # seleccione un pais. 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 estamos 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
  , 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 ad-stock. Seleccione el ad-stock
↪  geométrico para reducir tiempos de cómputo
)
print(InputCollect)
```

Ahora, obtén los nombres de los hiperparámetros que hay que ajustar:

```
hyper_names(adstock = InputCollect$adstock, all_media =
↪  InputCollect$all_media)
```

A continuación, defina los rangos en los que se moverán los hiperparámetros. Se utilizarán los mismos límites para α y γ que controlan la forma de la curva entre retorno decreciente y forma

de S, y el punto de inflexión respectivamente. En el caso de θ servirá para definir el ad-stock (recuerdo publicitario) que podríamos limitar teniendo en cuanta los intervalos recomendados por Meta: TV (entre 0,3 y 0,8), Resto medios off (entre 0,1 y 0,4) y Digital (entre 0 y 0,3). En nuestro caso, se dejará 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)
)
```

Ahora, añada los hiperparámetros al input que utilizaremos para ajustar los modelos:

```
InputCollect <- robyn_inputs(InputCollect = InputCollect, hyperparameters =
↪  hyperparameters)
print(InputCollect)
```

Y ejecute el modelo definiendo el número de iteraciones y trials. En nuestro caso empleará 2000 iteraciones y 10 trials:

```
OutputModels <- robyn_run(
  InputCollect = InputCollect,
  iterations = 2000,
  trials = 10,
  outputs = FALSE # desactive los outputs que se guardarán después
)
print(OutputModels)
```

Guarde ahora los resultados de la modelización 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 revisará los distintos modelos obtenidos y gráfico resumen que proporciona Robyn y ya tendrá listo un primer modelo que estima el impacto de cada uno de los medios publicitarios. Se profundizará en la interpretación de uno de los modelos obtenidos con buen ajuste. A través del onepager (figura 14.2), tendrá toda la información relativa al impacto de la publicidad, incluyendo métricas relativas a la bondad del ajuste como el R2 y el NRMSE en la parte superior:

Los principales outputs a evaluar están resumidos en cada uno de los gráficos del Onepager:

1. 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 todas las variables de medios).

2. Actual vs. Predicted Response: muestra el ajuste del modelo. En nuestro caso, la línea del ajuste (azul) y la real (naranja) se aproximan bastante, lo que muestra que ha sido capaz de explicar la mayor parte de la variabilidad de la serie de reservas.

3. Share of Spend vs Share of Effect with total CPA: muestra la relación entre la cuota de inversión y la cuota de la contribución generada por dichas inversiones. En nuestro caso, puede observar que tanto Online Video y Display están obteniendo mayor aporte que lo esperado por la inversión, lo que hace que el CPA (valores en azul) sean menores para estos medios que para el resto. La TV se encuentra en el polo opuesto y marca el máximo CPA, es decir, es el medio menos eficiente a la hora de generar reservas.

4. Response Curves and Mean Spends by Channel: muestra la relación no lineal existente entre las inveresiones y las reservas. Es el input principal a la hora de optimizar las inversiones.

5. Geometric Adstock: Fixed Decay Rate Over Time: muestra el efecto recuerdo (ad-stock) de cada medio publicitario. Puede observar que los medios offline (TV y el resto) son los medios que son capaces de generar un recuerdo más alto. No olvide que en la configuración del modelo podría haber fijado el intervalo en el que se tendría que mover el ad-stock de cada medio.

6. Fitted vs. Residual: muestra la dispersión entre los datos ajustados (eje x) y los residuos (eje y). Este gráfico debe mostrar que los puntos están aleatoriamente ubicados alrededor el eje horizontal.

Y, ¿cómo se pueden optimizar las inversiones en medios empleando estos resultados? Podrá utilizar las curvas obtenidas (parámetros guardados en *all_hyperparameters.csv*) para simular distintos escenarios y seleccionar el que dé máximas reservas. La otra opción sería utilizar la documentación de código incluida en el Step 5 para generar el reparto óptimo de inversión basado en un presupuesto fijado.

Figura 49.2: One pager de resultados de Robyn

En este capítulo se han dado los primeros pasos en la medición y optimización del ROI. Le animamos a que siga profundizando dentro de este campo tan apasionante y complejo para basar las decisiones futuras de inversión en análisis desarrollados en R.

```
sessionInfo()
```

Capítulo 50

Análisis electoral: de Rstudio a su periódico

Borja Andrino Turón

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.

50.1. 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 escrapeará 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.

```
require(ggplot2)
library(dplyr)
require(rvest)
require(lubridate)
require(sf)
```

```
require(ggtext)
require(rio)
require(janitor)
require(here)

load(here("data/data_periodismo.RData"))

url_provincias <-
  "https://www.ine.es/daco/daco42/codmun/cod_provincia.htm"

cod_provincias <-
  read_html(url_provincias) |>
  html_nodes("table.miTabla") |>
  html_table() |>
  purrr::map_df(as_tibble) |>
  rename(codigo_prov = 1, name_prov = 2) |>
  dplyr::mutate(codigo_prov = stringr::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 = stringr::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())
```

50.2. 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) |>
  dplyr::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) |>
  dplyr::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 |>
  dplyr::mutate(region = "Andalucía") |>
  group_by(convocatoria, region, bloque) |>
  summarise(votos_bloque = sum(votos_partido), .groups = "drop")

datos_bloques_provs <-
  datos_elecciones |>
  dplyr::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) |>
  dplyr::mutate(votos_bloque_pc = votos_bloque / validos)
```

A continuación, se realiza el gráfico con los daton 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 = scales::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))
  )
```

Replicar este mismo gráfico para cada provincia no es complicado. Solo 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.

Figura 50.1:
(#fig:plot-voto-andalucia)

```
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 = scales::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 50.2:
(#fig:plot-voto-andalucia-pro)

Este segundo gráfico 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) |>
  dplyr::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) |>
  dplyr::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.

Evolución de voto a partidos de derecha, izquierda
y otros desde 2015
2015
2018
2022
Fuente: Junta de Andalucía

Figura 50.3:

Capítulo 51

Evolución del paro registrado en Castilla-La Mancha

Isidro Hidalgo Arellano y *Ángel Jiménez Rojas*
Observatorio del Mercado de Trabajo de Castilla-La Mancha

51.1. Planteamiento

En los últimos 15 años el mundo ha sufrido dos grandes períodos de **crisis económicas**: 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 (Toharia, 2012) 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.
Castilla-La Mancha 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 han tomado las siguientes variables: **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 desagregado según estas variables a lo largo de los años que van desde 2007 a 2022 (éste último con datos desde enero hasta agosto).

Para este análisis se usan las librerías y variables (paletas de colores para los gráficos) siguientes:

```
library(CDR, tidyverse, ggpubr)
paleta_heatmaps <- c(rgb(.7, 1, 0, .5), rgb(.13, .22, .58, 1))
paleta_lineas <- c("blue4", "orange", "darkgreen")
```

Se cargan los datos, y se muestra la estructura de la `tibble` :

```
data("parados_clm")
parados_clm
# A tibble: 92,095 × 8
#    anyo  sexo    edad sector t_bus_e   tramo_edad t_bus_e_agr parados
#    <ord> <fct>  <dbl> <fct>  <ord>     <ord>      <ord>         <dbl>
#  1 2022  hombre    16 SINACT t<=7 días <30 años   t<=6 meses    7
#  2 2022  mujer     16 SINACT t<=7 días <30 años   t<=6 meses    8.25
#  3 2019  hombre    28 INDUST t<=7 días <30 años   t<=6 meses    8.58
#  4 2019  mujer     28 INDUST t<=7 días <30 años   t<=6 meses    3.83
#  5 2017  hombre    48 CONSTR t<=7 días >44 años   t<=6 meses   15.8
# ... with 92,085 more rows
```

51.2. Evolución del número total de parados

Para ver la evolución del número total de parados se puede construir un gráfico de evolución:

```
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
```

Figura 51.1: Evolución del paro registrado en CLM

De un primer análisis visual de la Fig. 51.1 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.

51.3. Evolución de la edad y el sexo en la población parada

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 51.2: Paro registrado según edad y sexo en 2007, 2019 y 2022

Se puede apreciar en la Fig. 51.2 que en los dos procesos críticos se ha producido un **desplazamiento a mayor edad**, siendo este cambio más pronunciado en las **mujeres**.
El mapa de calor es muy útil para ver inicialmente de forma visual estos cambios, pero si se

desea ver detalladamente cómo ha cambiado la distribución del paro según el `sexo` y la `edad`, se programa otra función, que proporciona mayor nivel de detalle:

```
densidad_compara <- function(variab, inicio = 2007, medio = 2019,
                             fin = 2022) {
  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)
}
```

La función `densidad_compara()` produce un cuadro de gráficos comparando las distribuciones de la edad dentro de cada estrato de la variable elegida, para tres años diferentes (2007, 2019 y 2022 por defecto). Ejecutando esta función para la variable `sexo` se obtiene:

```
densidad_compara("sexo")
```

Se observa en la Fig. 51.3 que las distribuciones del año 2007, anteriores a ambas crisis, que los hombres parados presentan **dos máximos**, en torno a 25 y 60 años, mientras que las mujeres tienen una distribución bastante centrada entre 30 y 40 años. En cambio, si se observan las distribuciones en 2022, se aprecia el desplazamiento de la distribución de ambos sexos a edades **mayores de 50 años**. Este desplazamiento es algo más intenso en las mujeres.

Figura 51.3: Distribución por edad y sexo para 2007, 2019 y 2022

Se observa igualmente que comparando las distribuciones de las mujeres de 2019 y 2022 la crisis de la COVID-19 ha desplazado 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.

51.4. Evolución del tiempo de búsqueda de empleo en la población parada

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 se toma como simplificación una agregación de cuatro tramos de tiempo de búsqueda de empleo (de los doce originales), ejecutando la función `densidad_compara()` para la variable `t_bus_e_agr` se obtiene:

```
densidad_compara("t_bus_e_agr")
```

En la Fig. 51.4 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, señalando que después de la crisis financiera de 2008 es más difícil encontrar empleo para una persona en **paro de larga duración**. Se puede afirmar también que los dos períodos de crisis han supuesto la creación de un **paro estructural de larga duración**.

Dejamos 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 correspondiente al tiempo de búsqueda de empleo **mayor de 24 meses** es más profundo en el colectivo de las mujeres. El código a utilizar es:

Figura 51.4: Distribución del paro registrado por edad y tiempo de búsqueda de empleo

```
heatmap_anyos("sexo", "t_bus_e")
```

51.5. Evolución del paro registrado 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 como "sin actividad". A la hora de analizar esta variable, por tanto, es importante tener en cuenta que una parte de los parados calificados como "sin actividad", realmente tuvieron un trabajo hace mucho tiempo.
Si se desea visualizar los cambios producidos en estas variables con un mapa de calor, es posible ejecutar de nuevo la función `heatmap_anyos()` para obtener:

```
heatmap_anyos("sexo", "sector")
```

Destaca en la Fig. 51.5 el incremento de paro registrado correspondiente al sector **servicios**, siendo este efecto mayor para las mujeres.
Ejecutando la función `densidad_compara()` para la variable `sector` se obtiene:

```
densidad_compara("sector")
```

Como se observa en la Fig. 51.6 las diferencias a lo largo del tiempo de las distribuciones 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 51.5: Paro registrado según sexo y sector de procedencia

Figura 51.6: Distribución del paro registrado 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** presenta igualmente desplazamiento en edad, pero además ha subido intensamente 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.

51.6. 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 52

Valoración del precio de la vivienda

David Rey y Pelayo Arbués

52.1. Introducción

Este capítulo construye un modelo de valoración de la vivienda utilizando se utiliza el paquete de datos **idealista18** que contiene datos de precios de vivienda plurifamiliar para las ciudades españolas de Madrid, Valencia y Barcelona (Rey et al., 2022). Como se ha indicado en capítulos anteriores del libro este conjunto de datos se ha puesto a disposición de la comunidad investigadora como un conjunto de datos abierto.

La valoración de viviendas es particularmente difícil la naturaleza heterogénea de este tipo de bien, que es consecuencia de los distintos atributos que la conforman, su estado de conservación y su ubicación. Para resolver esta cuestión, se puede utilizar la técnica de estimación de precios, denominada hedónica, que permite controlar los cambios de precio en función de su calidad, características o localización. La teoría de precios hedónicos calcula el precio de un bien mediante la agregación de las contribuciones de los diferentes elementos que lo componen. Dicho de otro modo, un piso vale más o menos en función de sus metros, es decir más metros suponen una utilidad positiva para el comprador. Pero no solamente existen contribuciones positivas por características físicas, sino que, un piso frente a la playa tiene mayor valor que otro situado en una zona interior de la ciudad. Esta utilidad se puede denominar espacial o de localización y está relacionada con la existencia de heterogeneidad espacial, entendida en este caso como la variación sistemática de los precios hedónicos a través de la geografía.

En la literatura no existe un consenso general sobre cómo especificar las covariables de localización, por lo que a menudo se construyen de manera arbitraria. Una mala especificación espacial conduce a sesgos asociados a varios fenómenos, que son: la heterogeneidad espacial y

la dependencia espacial[1] (LeSage and Pace, 2009).

En este capítulo se presenta una especificación de modelo de tipo hedónico que incluye distintas características de la vivienda a la vez que trata de modelar la naturaleza espacial de los datos utilizando efectos aleatorios espaciales sobre una superficie continua.

52.2. Conjunto de datos

```
# Se recomienda gestionar las dependencias con la librería renv
# https://rstudio.github.io/renv/articles/renv.html
library(INLA)
library(dplyr)
library(sp)
library(sf)
library(tmap)
library(knitr)
library(gt)
library(idealista18)
```

Se utilizan con dos sistemas de referencia espacial, el original que es *EPSG:4326* y con el que se trabajará, que son coordenadas cartesianas proyectadas en *EPSG:25830*.

```
projcrs_src <- "EPSG:4326"
projcrs_dest <- "EPSG:25830"
```

Se cargan los polígonos de las secciones de Madrid. Se añade una nueva variable denominada *district*, que tiene el código de cada distrito a partir del código de sección censal, sobre la que se seleccionan tres distritos sobre los que se trabajará: Centro (01), Arganzuela (02) y Retiro (03).

```
# Carga de areas censales para Madrid, INE 2011 (fuente en formato WKT CRS
↪  EPSG:4326)
# El código censal español se compone de 10 dígitos, compuesto por 2 dígitos
↪  para provincia,
# 3 para municipio, 2 para distrito y 3 para la sección censal.
polygons_2011 <- read_delim("data/polygons-2011-madrid.csv",
  delim = ";"
) |>
  st_as_sf(wkt = "WKT", crs = projcrs_src) |>
  st_transform(projcrs_dest) |>
```

[1] La heterogeneidad espacial existe si las viviendas tienen precios parecidos a viviendas cercanas, y la dependencia espacial cuando el precio de una vivienda afecta a los precios de su entorno.

```
  dplyr::mutate(district = substring(CUSEC, 6, 7))

# Seleccionamos los datos de la zona central
district_list_center <- c("01", "02", "03")
polygons_2011 <- polygons_2011 |>
  dplyr::filter(district %in% district_list_center)
```

En la Figura se muestran las secciones censales y los distritos del análisis.

```
plot(polygons_2011[, c("district")], main = "")
```

Figura 52.1: Secciones censales para los tres distritos de trabajo, cada color indica un distrito.

Se toman solo los inmuebles dentro de la zona central de la ciudad, previo haber reproyectado los datos al CRS de trabajo (*EPSG:2583*). Se usan solamente datos del último trimestre de 2018 (condición *PERIOD==201812*).

```
sample <- idealista18::Madrid_Sale |>
  st_transform(projcrs_dest) |>
  filter(PERIOD == 201812) |>
  group_by(ASSETID) |>
  filter(row_number() == 1) |>
  ungroup() |>
  st_join(polygons_2011, left = F)
```

52.3. Estimación del Modelo

Este ejemplo estima un modelo espacial de efectos aleatorios espaciales sobre una superficie continua. Para ello se utiliza un método de estimación bayesiana, dadas sus propiedades en cuanto a la interpretabilidad de los modelos y la medida de la incertidumbre de las estimaciones. Para reducir el coste computacional de estimar estos modelos, se utiliza Integrated nested Laplace approximation, INLA (Rue et al., 2009; Martins et al., 2013), un método computacional para la inferencia Bayesiana que es muy eficiente con modelos complejos y conjuntos de datos grandes. Este ejercicio se basa, en parte, del ejemplo del capítulo *"Spatial Models using Stochastic Partial Differential Equations"* del libro (Gómez-Rubio, 2020). En concreto el método aquí descrito está basado en la aproximación *SPDE* que se describe de forma detalla en (Bakka et al., 2018) que ajusta un modelo con un efecto espacial continuo.

Este modelo espacial se implementa como un efecto latente del tipo *SDPE* para INLA, este modelo requiere la construcción de una malla sobre la región de estudio, que se utilizará para aproximar el proceso espacial. Krainski (2018) describe en detalle los procesos *SDPE*. El proceso comienza por la construcción de una malla de tipo rectangular de 100 x 100 celdas, sobre las que se calcularán las predicciones del modelo. Es importante no confundir esta malla con la malla del proceso espacial.

```
# Combinamos todos los polígonos
madrid_comb <- st_combine(polygons_2011)
# Construimos la malla de 100 x 100 rectángulos sobre la ciudad
madrid_grid <- st_make_grid(madrid_comb, n = c(100, 100))
```

Dado que esta malla es más amplia que la región de estudio, se recorta para contener solamente las celdas circunscritas en los polígonos de la ciudad.

```
# Seleccionamos solo las celdas que estén dentro de un polígono
aux <- st_intersects(madrid_grid, polygons_2011)
madrid_grid <- madrid_grid[unlist(lapply(aux, length)) > 0]

# Para ver la malla descomentar la siguiente linea
# plot(madrid_grid, asp = 1)
```

El siguiente paso es definir la malla sobre la que se va a ajustar el modelo *SPDE*. La malla es en realidad un sistema de coordenadas discreto sobre el que se estimará el modelo, donde los puntos en el espacio sobre los que se trabajan son los vértices del proceso de triangulación que calcula la función `inla.mesh.2d`, para más información sobre este proceso véase (Bakka et al., 2018).

```
m_coords <- st_coordinates(sample)

# Creación de malla
mesh <- inla.mesh.2d(
```

```
  loc.domain = m_coords,
  max.edge = c(550, 1200),
  offset = c(600, 1200)
)

# Para ver la malla que usaremos en el hedónico descomentar las siguientes
↪  lineas
# plot(mesh)
# points(m_coords, pch = 19)
```

El modelo bayesiano establece la distribución a-priori para los parámetros de función de covarianza *Mátern* que definen el efecto aleatorio espacial. Cuando se construye la matriz, se establecen los parámetros de rango y σ.

```
# Creación de SPDE
madrid.spde <- inla.spde2.pcmatern(
  mesh = mesh, alpha = 2,
  prior.range = c(2, 0.95), # P(range < 2) = 0.95
  prior.sigma = c(10, 0.01)
) # P(sigma > 10) = 0.01
```

La matriz A se utiliza para calcular el efecto aleatorio para los vértices de la malla generada a través del proceso de triangulación anterior, que se corresponden con la situación real de las distintas viviendas del conjunto de datos. El efecto aleatorio espacial de cada vivienda se calcula como el peso de la suma de los efectos aleatorios de los vértices entre los que está cada vivienda (Bakka et al., 2018).

```
# Creación de la matriz proyectora A
a_madrid <- inla.spde.make.A(mesh = mesh, loc = m_coords)
```

El objeto *s.index* contiene el índice para el modelo espacial gaussiano latente mediante `inla.spde.make.index()`. Se deben especificar el nombre del índice s y el número de vértices en el modelo SPDE *spde$n.spde*.

```
s.index <- inla.spde.make.index(
  name = "spatial.field",
  n.spde = madrid.spde$n.spde
)
```

Para introducir los datos en el modelo INLA, es necesario definir una estructura de dato de tipo *stack* que incluya los distintos parámetros: los efectos aleatorios y los efectos fijos. La variable dependiente será el precio de venta en €/m².

```
madrid.stack <- inla.stack(
  data = list(UNITPRICE = sample$UNITPRICE),
  A = list(a_madrid, 1),
  effects = list(
    c(s.index, list(Intercept = 1)),
    list(
      ROOMNUMBER = sample$ROOMNUMBER,
      BATHNUMBER = sample$BATHNUMBER,
      HASLIFT = sample$HASLIFT,
      district = sample$district,
      CONSTRUCTEDYEAR = sample$CONSTRUCTIONYEAR,
      CADASTRALQUALITYID = sample$CADASTRALQUALITYID
    )
  ),
  tag = "madrid.data"
)
```

Además de la información para ajustar el modelo, se desea estimar el efecto espacial aleatorio (más el intercepto del modelo) sobre un conjunto de puntos definidos mediante una malla rectangular *grid*, para cada una de las celdas se toma el centroide sobre el que se calculará la predicción. Para realizar la inferencia se debe crear un nuevo un nuevo *stack* (variable *madrid.stack.pred*).

Para poder hacer la inferencia se deben estimar las variables independientes, como los puntos del grid no se corresponden con una vivienda se imputan los valores de calidad constructivas y año de construcción medianos de su sección censal.

```
# Medidas por sección censal
sample_cusec <- sample

# Eliminamos la geometría para hacer que la agregación sea más rápida
st_geometry(sample_cusec) <- NULL

sample_cusec <- sample_cusec |>
  dplyr::group_by(CUSEC) |>
  dplyr::summarise(
    sscc_constructedyear = median(CONSTRUCTIONYEAR, na.rm = T),
    sscc_cadastralqualityid = median(CADASTRALQUALITYID, na.rm = T),
    sscc_district = first(district)
  )

polygons_2011 <- polygons_2011 |>
  dplyr::inner_join(sample_cusec, by = "CUSEC")

# Convertimos el objeto sfc a sf
```

```
# Tomamos solo los centroides para que solo haya una correspondencia
#  con la seccion censal

grid_attributes <- st_as_sf(st_centroid(madrid_grid)) |>
  st_join(polygons_2011)
```

La inferencia se realiza sobre pisos de dos habitaciones, 1 baño y con ascensor, que tengan las mismas características constructivas (año de construcción y calidad constructivas) que las de su sección censal.

```
grid_coords <- st_coordinates(st_centroid(madrid_grid))

# Creamos la estructura de datos para la predicción
A.pred <- inla.spde.make.A(mesh = mesh, loc = grid_coords)
madrid.stack.pred <- inla.stack(
  data = list(UNITPRICE = NA),
  A = list(A.pred, 1),
  effects = list(
    c(s.index, list(Intercept = 1)),
    list(
      ROOMNUMBER = 2,
      BATHNUMBER = 1,
      HASLIFT = 1,
      district = grid_attributes$district,
      CONSTRUCTEDYEAR = grid_attributes$sscc_constructedyear,
      CADASTRALQUALITYID = grid_attributes$sscc_cadastralqualityid
    )
  ),
  tag = "madrid.pred"
)
```

Posteriormente, se unen ambas estructuras de tipo *stack* para su uso en el ajuste y predicción.

```
join.stack <- inla.stack(madrid.stack, madrid.stack.pred)
```

En la especificación del modelo se utilizan efectos aleatorios, efectos fijos y el efecto espacial aleatorio del modelo *spde*. El ejercicio pretende mostrar la construcción de un modelo bayesiano espacial en el que intervienen además otras covariables como efectos fijos y efectos aleatorios. Los efectos fijos se asocian al número de habitaciones, baños y ascensor. Los efectos aleatorios, a la combinación entre distrito y calidad constructiva. El modelo, por tanto, se especifica de la siguiente manera:

```
form <- log(UNITPRICE) ~ -1 + Intercept +
  ROOMNUMBER +
  BATHNUMBER +
  HASLIFT +
  f(district, CADASTRALQUALITYID, model = "iid") +
  f(spatial.field, model = spde)
```

Como se puede observar, se utilizan como variables explicativas del modelo hedónico el número de habitaciones, número de baños y si la finca cuenta con ascensor. Estas variables se introducen como efectos fijos en el modelo lo que significa que este efecto se calcula de forma general para todo el ámbito de estudio, la ventaja de utilizar efectos fijos es interpretabilidad, dado que es conocida la distribución de cada uno de los coeficientes. La desventaja es que se asume que la contribución del número de los efectos fijos es constante para toda la ciudad, lo que no es necesariamente cierto. Este supuesto puede relajarse en aquellos casos que el investigador considere conveniente, introduciendo un efecto aleatorio asociado a la variable oportuna.

Por otra parte se ha incluido un término de efectos aleatorios asociados a distrito y calidad constructiva, el motivo de hacerlo así es equivalente a el uso de "*dummies* de zonas" con un desglose por calidad de la construcción. Por tanto este modelo capturará el efecto de estas dos variables en todo el conjunto, estima los efectos fijos de los atributos de los inmuebles y ajusta a través del modelo espacial la contribución de los precios debida a la ubicación. Por este motivo se puede considerar en este modelo que la contribución espacial es compartida por el modelo *spde* y por los códigos de distrito. El modelo espacial puede extenderse para incorporar el aspecto temporal, para más información ver (Moraga, 2019).

Para ajustar el modelo se utiliza la función `inla`, y dado que, se incluye en el *stack* el set para la inferencia, se calculan los datos para la malla rectangular definida.

```
m1 <- inla(form,
  data = inla.stack.data(join.stack, spde = madrid.spde),
  family = "gaussian",
  control.predictor = list(
    A = inla.stack.A(join.stack),
    compute = TRUE
  ),
  control.compute = list(cpo = TRUE, dic = TRUE)
)
```

A continuación, se muestra un descriptivo del modelo ajustado, por una parte se tiene el valor de la intersección (*INTERCEPT*) y los coeficientes asociados a los efectos fijos. La salida completa del modelo se puede obtener mediante el comando `summary(m1)`. Se observa que el coeficiente para *ROOMNUMBER* toma un valor negativo, debido a que, a igualdad de condiciones, el precio por unidad de superficie es mayor en pisos pequeños que en pisos grandes. En los casos de *BATHNUMBER* y *HASLIFT* se observan contribuciones positivas al precio unitario de la vivienda.

El descriptivo también muestra el efecto aleatorio asociado a la combinación entre distrito y calidad constructiva, y al efecto espacial. Los valores de la medida DIC recogen una medida de la calidad del modelo[2].

Estimaciones del Modelo

	mean	sd	0.025	0.5	0.975
Fixed Effects					
Intercept	8.512	0.023	8.467	8.512	8.557
ROOMNUMBER	-0.061	0.003	-0.066	-0.061	-0.055
BATHNUMBER	0.054	0.005	0.046	0.054	0.063
HASLIFT	0.080	0.007	0.066	0.080	0.094
Random Effects					
Precision for the Gaussian observations	19.594	0.355	18.924	19.583	20.322
Precision for district	17857.142	6887.207	8.557	256.484	67612.838
Range for spatial.field	32.973	61.229	4.421	16.456	164.119
Stdev for spatial.field	3.999	3.170	0.315	3.185	11.852

Deviance Information Criterion (DIC): -751.93

Para mostrar el efecto aleatorio (más la intersección), se toma la media de la posterior de la suma de los valores ajustados. Sobre las predicciones en *madrid.pred*, se crea un *dataframe* con la media y la desviación estándar transformándolos a la escala original de precios por superficie.

```
index.pred <- inla.stack.index(join.stack, "madrid.pred")$data

tab <- data.frame(
  SPDE = exp(m1$summary.fitted.values[index.pred, "mean"]),
  SPDE_SD = exp(m1$summary.fitted.values[index.pred, "sd"])
)
# Creamos objecto sf
madrid_grid2 <- st_sf(data = tab, geom = madrid_grid)
```

A continuación, se muestran las medias y desviaciones estándar de las distribuciones posteriores. Se puede observar como el modelo captura el efecto espacial de los precios en la ciudad de Madrid.

```
plot(madrid_grid2["data.SPDE"],
  main = " "
)
```

Conocer la distribución espacial de la desviación estándar es útil para conocer dónde las estimaciones del modelo tienen mayor incertidumbre. Habitualmente la cuestión está relacionada con la disponibilidad de información en la zona, en la figura se observa como las zonas del norte de la ciudad, donde hay menos viviendas en venta, tienen un mayor nivel de variabilidad, al contrario de lo que sucede en torno al centro (Sol) donde la variabilidad es menor.

[2]La medida DIC es una generalización del Criterio de Información de Akaike (AIC), son particularmente útiles en modelos bayesianos y representan el compromiso entre la bondad del ajuste y complejidad del modelo.

Figura 52.2: Media posterior

```
plot(madrid_grid2["data.SPDE_SD"], main = " ")
plot(madrid_comb, add = TRUE)
```

Figura 52.3: Desviación estándar modelo INLA

Como se observa en los resultados obtenidos, la estimación de modelos hedónicos usando aproximaciones bayesianas es muy interesante dado que permite estimar distribuciones de los parámetros de los modelos así como de las predicciones. De esta forma, los resultados permiten analizar la precisión de las estimaciones. Por otro lado, el empleo del método de computación INLA permite la aplicación de modelos complejos en número de parámetros a conjuntos de

datos grandes.

Capítulo 53

Segmentación de clientes en retail

Jaime Fierro Martín, Rocío González Martínez y Cristina Sánchez Figueroa

53.1. 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. 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 compañia, 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.

53.2. Del Modelo RFM tradicional al Modelo RFM extendido, mejoras propuestas

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 mometario, **(*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. Con este sencillo ejemplo comprobamos que los enfoques de conjuntos clásicos pueden resultar poco funcionales (Martínez et al., 2019). Por ello, se propone una mejora en la definición de los intervalos mediante la aplicación del **ranking de percentiles**. Este método 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**.

Ranking de percentiles cliente $x_i = \frac{Número\ de\ valores\ inferiores\ a\ x_i}{Número\ total\ de\ clientes}$

53.3. Modelo RFM extendido

Los autores de este caso práctico recomiendan seguir una metodología de gestión de proyectos. La **Metodología CRISP-DM**(Chapman et al., 2000b) es un estándar ampliamente utilizado en los proyectos de ciencia de datos.

Una vez definido el problema , la recopilación y comprensión de los datos se establece como etapa esencial para el desarrollo del proyecto.

53.3.1. 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.

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, a modo meramente ilustrativo mencionamos RSQLite, odbc, mogolite, httr, entre otras muchas.

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
#> 4 num_2113698 2021-08-27 37.2 id_0814765
#> 5 num_0794486 2020-11-21 44.2 id_0494840
#> 6 num_0184609 2021-05-22 12.0 id_0214173
```

53.3.2. Cálculo de las variables del Modelo RFM

Identificadas las variables iniciales, es necesario calcular las variables del Modelo RFM:

- La variable actualidad, *recency* (R), es el intervalo de tiempo transcurrido desde la compra más reciente por 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 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
```

53.3.3. Breve análisis exploratorio de las variables del Modelo RFM

Del análisis de las variables obtenidas puede concluirse que:

- 107 929 clientes únicos 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*.
- 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 fijada del 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))
```

53.3.4. 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.

Figura 53.1: Box-plot de variables RFM

```
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
```

```
head(rfm_rank) # el lector puede ver la puntuación del ranking
```

53.3.5. 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 k-means 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* 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 *Elbow method, Average silhouette method y Gap statistic method*, que gracias a la función `fviz_nbclust()` del paquete `factoextra` se pueden calcular con facilidad para realizar una buena elección.

El número óptimo de *clusters* se calcula:

```
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 53.2: Número óptimo de clusters

Mediante el *método Elbow* 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 pertence el
↪  cliente
```

53.3.6. Descriptivos e interpretación de los segmentos

Los segmentos obtenidos deben tener sentido y utilidad de negocio. Para ello es imprescindible realizar un el descriptivo de cada segmento y proceder a su interpretación de 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' que fueron buenos clientes en términos monetarios y de frecuencia pero que hace tiempo que no realizan nuevas compras. La compañía debe dedicar esfuerzos en recuperar estos clientes para convertirlos al segmento TOP.
- 3-*Top*: reúne 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(
    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"
)
```

Figura 53.3: Box-plot de variables RFM

53.3.7. Puesta en producción

Calculado el modelo RFM k-means la compañía puede incorporan 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()` del paquete en `R clue: Cluster Ensembles` facilita realizar la actualización periódica de los segmentos con el modelo entrenado.

Capítulo 54

Ciencia de datos para la investigación y ensayos clínicos

Autores: *Alberto M. Borobia y María Jiménez-González*

54.1. 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 diagnósticados en España arrojaban un diagnóstico claro: se necesita más investigación.

El objetivo de este capítulo es construir un esquema sobre el uso de datos en investigación clínica y ensayos clínicos a través de teoría y ejemplos prácticos.

54.2. Introducción al uso de datos en investigación clínica y ensayos clínicos

54.2.1. ¿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 (Cruz Rivera et al., 2020).
- El PAE detalla las características principales del eventual análisis estadístico de los datos 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" (ICH).

54.2.2. Limitaciones de los estudios observacionales

En el apartado anterior, se pone 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 la introducción del *sesgo*. Los ensayos clínicos tienen como principal objetivo eleminar 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).

54.2.3. Propensity Score

Una solución aconsejable y recomendada ante los sesgos "escondidos" en los estudios observacionales es la técnica *propensity Score* o la puntuación de propensión. Esta técnica de emparejamiento equilibra las covariables observadas sesgadas ajustando por su puntuación de propensión, eliminando presumiblemente el sesgo.

54.3. Ejemplo práctico 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 determinar los factores de riesgos asociados a la mortalidad causada por esa enfermedad.

```
library(readxl)
datos_observacional <- read_excel("data/datos_observacional.xlsx")
head(datos_observacional, 5)
#> # A tibble: 5 x 8
#>      ID fecha_hospitalizacion sexo    edad comorbilidades fecha_alta exitus
#>   <dbl> <dttm>                <chr>  <dbl> <chr>          <chr>       <dbl>
#> 1     1 2015-04-17 00:00:00   Mujer     76 1 o más        17/04/2020      1
#> 2     2 2015-03-21 00:00:00   Mujer     64 1 o más        31/03/2020      0
#> 3     3 2015-04-09 00:00:00   Hombre    65 1 o más        16/04/2020      0
#> 4     4 2015-04-04 00:00:00   Hombre    77 1 o más        13/04/2020      0
#> 5     5 2015-03-24 00:00:00   Mujer     66 1 o más        27/03/2020      0
#> # ... with 1 more variable: fecha_exitus <dttm>
```

Anteriormente, se ha evidenciado que las mujeres de mayor edad y con 1 o más comorbilidades tienen más riesgo de fallecer por la enferdad **X**.

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 de estudio. El paquete *tableone* integra funciones específicas para la creación de esas tablas. La función principal de este paquete es `CreateTableOne()`.

```
library(tableone)

myVars <- c("sexo", "edad", "comorbilidades")
nonnormal <- c("edad")
factorVars <- c("sexo", "comorbilidades")
tab1 <- CreateTableOne(
  vars = myVars, factorVars = factorVars,
  strata = "exitus", data = datos_observacional
)
```

Muestra la tabla con código **R** (se omite la salida por simplicidad)

```
tab1 <- print(tab1,
  showAllLevels = TRUE, formatOptions = list(big.mark = ","),
  exact = "stage", nonnormal = nonnormal
)
```

Presenta la tabla de resultados formateada:

```
knitr::kable(tab1,
  caption = "Características basales de la población"
)
```

Tabla 54.1: Características basales de la población

	level	0	1	p	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	43 (54.4)	18 (85.7)	0.018	
	No	36 (45.6)	3 (14.3)		

En la Tabla **??**, se confirma el sesgo y el desequilibrio existente en las variables *sexo, edad y comorbilidades*, por lo que se procede al ajuste de la técnica *propensity score*.

El paquete `MatchIt` integra las funciones principales para el ajuste de la técnica:

- 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 de la puntuación de propensión.
- La función `get_matches`, empareja 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
)
datos_observacional <- get_matches(match, datos_observacional)
```

Para comprobar que el sesgo evidenciado en estudios anteriores ha desaparecido, es posible reproducir la Tabla **??**.

```
tab2 <- CreateTableOne(
  vars = myVars, factorVars = factorVars,
  strata = "exitus", data = datos_observacional
)

tab2 <- print(tab1,
  showAllLevels = TRUE, formatOptions = list(big.mark = ","),
```

Tabla 54.2: Características basales de la población emparejadas

	level	0	1	p	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	43 (54.4)	18 (85.7)	0.018	
	No	36 (45.6)	3 (14.3)		

```
  exact = "stage", nonnormal = nonnormal
)
```

```
knitr::kable(tab2,
  caption = "Características basales de la población emparejadas"
)
```

54.3.1. 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.

El *dataset* está formado por 301 pacientes, 101 diagnósticados con infección por SARS-CoV-2 y 100 exitus.

```
library(readxl)
datos_supervivencia <- read_excel("data/datos_supervivencia.xlsx")
head(datos_supervivencia, 5)
#> A tibble: 5 × 7
#> id EXITUS_TIME DIAG_COVID EXITUS N_COMORBIDITIES SEX     EDAD
#>  <dbl>       <dbl>      <dbl>  <dbl>       <dbl> <chr>  <dbl>
#> 262           0          1      1               5 Hombre    83
#> 236           1          1      1               5 Hombre    72
#> 170          11          0      0               2 Mujer     65
#> 204          11          1      1               4 Hombre    80
#>  46          14          1      1               5 Hombre    90
```

54.3.2. Estimación y comparación de curvas de supervivencia

La función (o curva) de supervivencia estudia la probabilidad de que la variable, en este caso 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 para el ajuste de la técnica:

- La función `Surv()`, crea un objeto de supervivencia formado por el tiempo hasta la ocurrencia del evento y el evento (exitus).
- La función `survfit`, 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*.
  - `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 del estudio.

```
library(survival)
library(survminer)
```

```
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"),
  # añade tabla de supervivencia
  risk.table = TRUE,
  tables.height = 0.2,
  tables.theme = theme_cleantable(),
)
```

Figura 54.1: Curva de supervivencia XXXXXXX

54.3.3. Regresión de COX

La regresión de Cox 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). Es una técnica semi-paramétrica ya que no asume *a priori* ningún modelo de supervivencia pero sí en los efectos de las variables predictoras.

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.

```
fitCOX <- coxph(Surv(EXITUS_TIME, EXITUS) ~ DIAG_COVID + EDAD + SEX +
↪  N_COMORBIDITIES,
  data = datos_supervivencia
)
```

El *output* principal de una regresión de Cox son los *Hazard Ratio* (HR), que es la interpretación de la exponencial de los coeficientes de regresión del modelo. Esta razón de riesgos oscila entre 0 a ∞ , siendo el intervalo [0,1] una relación de riesgo menor y [1, ∞] una relación de riesgo mayor.

Nota para facilitar la interpretación:

- Los HR localizados entre [1,2] se interpretan en porcentaje. Es decir, HR = 1.5 es igual a un aumento del riesgo del 50 %.
- Los HR localizados entre [2, ∞] se interpretan en "veces". Es decir, HR = 3 es igual a un aumento del riesgo de 3 veces.
- Los HR localizados entre [0,1] es recomendable hacer su inversa. Es decir, HR = 0.8 es igual a una disminución del riesgo del 20 %.

```
summary(fitCOX)
#> 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
```

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 %.

54.4. Conclusión

Ha sido necesaria una pandemia mundial para que la sociedad empiece a dar visibilidad y reconocimiento no sólo a la bioestadística, si no 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, si no en el apoyo a la ciencia.

Es, quizá, la paradoja más extraña pero que representa el dicho popular:

La sociedad no avanzará si la ciencia no lo hace primero.

Capítulo 55

Lo que nos cuentan los datos sobre el cambio climático

Dominic Royé

La temperatura media global en la superficie ha aumentado en 1.1 ºC desde la era preindustrial (1880-1900), puede parecer poco pero implica un aumento significativo en el calor acumulado del sistema tierra. Cuando se combinan el aumento de de la temperatura con respecto a la superficie terrestre y el océano la tasa promedia es de 0,08 ºC por década desde 1880. Sin embargo, la tasa promedio de aumento desde 1981 ha sido más del doble, con 0,18ºC. 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 climático[1].

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 0,34ºC cada década desde principios de los 80 (Cramer et al. (2020)).

En este caso práctico con datos sobre el cambio climátio se tratan las anomalías de la temperatura superficial del mar Mediterráneo en los meses estivales desde el año 1982 a 2022. 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 proporciona la 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 rellenadas por interpolación (https://developers.google.com/earth-engine/datasets/catalog/NOAA_CDR_OISST_V2_1). El geoprocesamiento en nube está explicado en el capítulo XXX.

[1] https://www.ncei.noaa.gov/news/global-climate-202112

55.1. 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 (NetworkCommon 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 multivariables. Los datos multidimensionales en formato *geotiff* son menos comúnes, pero también se pueden llegar a usar en ocaciones. Además, es posible crear objetos multidimensionales importando multiples archivos ráster.

55.2. 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, del que hacen uso también otros paquetes aunque no se ve. El manejo con `{ncdf4}` es algo complejo, particularmente por la necesidad de gestionar la memoria RAM cuando tratamos grandes conjuntos de datos o también por la forma de manejar la clase *array*. Otro paquete muy potente es `{terra}`, que conocemos cuando trabajamos con datos ráster y permite usar sus funciones también para el manejo del formato netCDF.

```
# instalar los paquetes si hace falta
library("tidyverse")
library("rmapshaper")
library("giscoR")
library("sf")
library("lubridate")
library("terra")
library("fs")
library("patchwork")
library("scales")
library("rmapshaper")
library("RColorBrewer")
library("CDR")
```

55.3. Visualización de mapas de 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.

55.3.1. Datos

Importamos el polígono del Mar Mediterráneo para poder limitar los datos al área de interés.

```
data("med_limit")
```

A continuación se importa 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 cuando hayamos realizado el geoprocesamiento en nube de las anomalías (Capítulo X) podemos 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
```

55.3.2. Preparar los datos

Todavía se debe 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()`, la 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")
```

```
# plot
plot(anom)
```

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.

Todavía falta un cambio importante. Ahora mismo se tiene la misma variable, la anomalía, distribuida en muchas columnas, no obstante la estructura adecuada debe ser un conjunto total

Figura 55.1: Selección de anomalías 1982 a 1997 de los datos brutos.

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
  ))
```

Antes de pasar a construir el gráfico se estima la media de la anomalía global para toda la cuenca mediterránea de cada año. Estos datos se añadirá 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 reproyectamos.

```
# 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()
```

55.3.3. Construir el 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 empleamos `geom_tile()` o `geom_raster()` en *ggplot2*. La última geometría requiere una rejilla regular. Para este primer ensayo se filtra sólo el año 2003, 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()
```

Manteniéndose en 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 define 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. La idea es situar el texto a la derecha del punto, por eso, se hace un ajuste en longitud indicando un valor correspondiente en el argumento *nudge_x* en la unidad del sistema de coordenadas (SC). Recuerde que el SC está en metros.

La función `number()` del paquete `{scales}` facilita formatear las cifras con un decimal y el símbolo 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 = 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
  )) +
  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, una década. Si no se hiciese el *facet grid* empezaría por 1982 sin posibilidad de mantener en cada fila la década correspondiente. A todos los mapas se añadirán los límites de la cuenca mediterránea, no obstante, para que 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 todos 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 o/y 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 borramos 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 coor-

denadas 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()
```

En principio, se podría quedarse aquí y exportar el resultado, no obstante, lo que faltaría es un mapa de orientación.

55.3.4. Mapa de orientación

A través del paquete `{giscoR}` se obtiene los límites administrativos de los que únicamente se queda con una selección. También se limita la extensión a aproxidamente la de la cuenca mediterreánea. La función `ms_innerlines()` del paquete `{rmapshaper}` facilita la obtención de los límites compartidos o interiores de los países selecionados. En la construcción del mapa de orientación se incluye con ayuda de `geom_sf_text()` los nombres de los países en forma de código ISO-3.

```
# 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 inernos
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()
  )
```

55.3.5. Exportar mapa final

Se insertará el mapa de orientación en el gráfico como elemento adicional en la esquina derecha-arriba. El paquete `{patchwork}` puede ayudar a crear composiciones de distinos gráficos. Aquí empleamos la función `inset_element()` indicando 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
)
```

Figura 55.2: Anomalía estival de la temperatura de superficie del mar

Capítulo 56

Predicción de demanda eléctrica con deep learning

Jose Manuel Sanz Candales

56.1. 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 periodos temporales (anual, horario, quinceminutal).

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 la demanda eléctrica en España** para el año actual y el siguiente (2018 y 2019).

IMPORTANTE: Este desarrollo no estaba previsto en el presupuesto del año , por lo que tanto el software como los datos de entrada deben ser, a ser posible, gratuitos.

56.2. Datos de entrada

Respecto a los datos de entrada para el modelo, en este caso de uso será necesario utilizar un modelo regresor de aprendizaje supervisado, dado que la demanda eléctrica es una variable numérica contínua. Por ello, se necesitará una serie histórica de demanda eléctrica anual real en España de los años anteriores a 2018 y alguna/s variable/s predictora/s a partir de la/s cual/es los modelos de Machine Learning puedan encontrar relaciones para predecir con una precisión aceptable la demanda eléctrica.

Respecto a la serie histórica de demanda anual, Red Eléctrica, en su Web corporativa, publica datos estadísticos accesible 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 nuestros modelos. Se realiza una búsqueda de otras fuentes y, afortunadamente, se encuentra que el IDAE publica datos desde 1990 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 tendrá 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 la demanda eléctrica con la variable predictora que se va a utilizar.

Ya se tiene, por tanto, una parte de los datos necesarios para entrenar nuestros modelos de predicción, que es **la serie histórica de nuestra target (o variable a pedecir)**

La otra parte que se requiere son **las features o variables predictoras**. Se sabe que, históricamente, las variaciones interanuales de la demanda eléctrica dependen del comportamiento de la economía de una forma directa: si la economía crece, también crece la demanda (y viceversa). Se busca qué variables econométricas relevantes es posible encontrar de forma pública y, tras un detallado análisis, se concluye que la más adecuada para el modelo puede ser el PIB per cápita que se puede encontrar en el siguiente enlace, disponible de forma pública en Expansión - datos macro.

Ya se tiene el dataset de entrada para los modelos (como se verá a continuación, es un conjunto de datos muy pequeño y sencillo), y se puede comenzar a construir nuestro modelo en R.

56.3. Caso de estudio

Lo primero que se hará en la siguiente celda de código es leer el fichero que contiene el conjunto de datos, convertirlo de formato data.table a data.frame y visualizar sus primeras filas:

```
library(CDR)
df <- CDR::consumoelectricoanual
class(df) <- class(as.data.frame(df))
head(df, 3)
#> Año PIB Consumo
#> 1 1990 10.331 10.817
#> 2 1991 11.400 11.061
#> 3 1992 11.800 11.244
```

En este caso, ya se dispone de los datos reales de 2018 y 2019 (esto permitirá ver cómo ha sido la precisión del modelo), pero en un caso real, a principios de 2018 el PIB per cápita de 2018 y 2019 será una predicción y la demanda eléctrica anual es desconocida, ya que es lo que se necesita predecir). Es decir, los datos de demanda eléctrica de 2018 y 2019 para el modelo que

vamos a construir no existen, no se van a utilizar para entrenar ni para evaluar la precisión del modelo.

En la siguiente celda de código, se puede ver la correlación entre la variable predictora (PIB per cápita) y la variable a predecir (Demanda eléctrica anual):

```
cormat <- round(cor(df[c("PIB", "Consumo")]), 2)
head(cormat)
#> PIB Consumo
#> PIB 1.00 0.96
#> Consumo 0.96 1.00
```

Como se puede observar, el coeficiente de correlación lineal entre ambas variables es muy alto y positivo (es decir, cuando crece una también crece la otra). Si, además, se visualiza la gráfica de ambas variables en el tiempo, se apreciará de forma aún más clara:

```
library(ggplot2)
library(reshape2)
df_m <- melt(df, 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)
```

Hasta 2005 se observa que las líneas están prácticamente superpuestas, pero desde 2006 las líneas se separan, ¿a qué puede deberse este cambio de tendencia? Uno de los motivos más relevantes probablemente sean las medidas de eficiencia energética que se han ido introduciendo en los últimos lustros (iluminación led, electrodomésticos y 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 en dos partes: i. entrenamiento+validación (desde 1990 hasta 2017) y ii. test (2018 y 2019):

```
set.seed(123)
df_aux <- df[df$Año < 2018, ]
n <- nrow(df_aux)
trainIndex <- sample(1:n, size = round(0.9 * n), replace = FALSE)
df_train <- df_aux[trainIndex, ]
df_test <- df_aux[-trainIndex, ]
```

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 solo se probará un modelo, una Red Neuronal simple (Perceptrón Multicapa).

En realidad, no se va a entrenar un solo modelo sino varias Redes Neuronales, ya que se van a utilizar dos técnicas que son Grid Search (para probar distintas combinaciones de hiperparáme-

tros) y Cross Validation (para entrenar y validar aprovechando todos los registros del conjunto de entrenamiento-validación).

En este caso, se van a probar distintas redes combinando el número de capas ocultas y de neuronas por cada capa oculta:

```
library(neuralnet)
library(caret)
grid <- expand.grid(layer1 = c(5), layer2 = c(5), layer3 = c(2, 5))
grid
set.seed(123)
# define training control
train_control <- trainControl(method = "cv", number = 2, verbose = TRUE)
# train the model
model <- train(Consumo ~ PIB, data = df_train, trControl = train_control,
↪  method = "neuralnet", tuneGrid = grid)
```

Se muestran los resultados:

```
print(model)
#> Neural Network
#>
#> 25 samples
#> 1 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (2 fold)
#> Summary of sample sizes: 13, 12
#> Resampling results across tuning parameters:
#>
#> layer3 RMSE Rsquared MAE
#> 2 3.906921 0.9801918 3.5864138
#> 5 1.161004 0.9349297 0.9872479
#>
#> 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.
```

```
X <- df_test[c("PIB")]
#> Año PIB Consumo
#> 2 1991 11.40 11.061
#> 12 2001 17.20 17.279
```

```
#> 27 2016 23.98 19.993
predict(model, X)
#> 2 12 27
#> 11.48053 17.11479 21.11972
```

El modelo con mejor resultado (Red Neuronal con tres capas de 5 neuronas cada una) es conveniente volverlo a entrenar con el conjunto de entrenamiento-validación completo:

```
set.seed(122)
nn <- neuralnet(Consumo ~ PIB, data = df_train, hidden = c(5, 5, 5))
```

Ahora se predice con el modelo entrenado para la parte del conjunto de datos que se habían reservado para test (años 2018 y 2019):

```
pred <- predict(nn, X)
pred
#> [,1]
#> 2 11.61718
#> 12 16.94962
#> 27 21.75289
```

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 generaliza bien (esto es, no ha sobreentrenado ni subentrenado)

Ahora sí, se va a predecir el consumo eléctrico anual para 2018 y 2019, que es el dato que solicitaron desde el área de planificación de la empresa. Para ello, simplemente se utilizará el método predict del mejor modelo. Previamente, se va a añadir una columna en el conjunto de datos original -df- para luego 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(nn, df_pred_2018[c("PIB")])
```

Se hace también la predicción para el año 2019 y se visualizan las predicciones añadidas al conjunto de datos para ambos años:

```
df_pred_2019 <- df[df$Año == 2019, ]
df$Prediccion_MLP[df$Año == 2019] <- predict(nn, df_pred_2019[c("PIB")])
tail(df, 3)
#> Año PIB Consumo Prediccion_MLP
#> 28 2017 24.97 20.559 NA
#> 29 2018 25.75 20.504 20.78726
#> 30 2019 26.42 20.166 20.89161
```

Estos son los datos que se entregarían como resultado de la petición de información (convertidos a MWh aplicando el coeficiente que se citó anteriormente).

Claro está, en el momento que se entregan las predicciones para 2018 y 2019 todavía no se sabría cómo de buenas han sido, pero a principios de 2020 sí es posible calcular los errores de nuestro modelo, y es lo que se hará en las siguientes chunks:

```
library(reshape)
df_m_mlp <- melt(df, id.vars = "Año")
# options(repr.plot.width=15, repr.plot.height=8)
ggplot(df_m_mlp, aes(Año, value, col = variable)) +
  geom_point(size = 3)
```

Figura 56.1: xxxxx

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.Parece que el modelo ha tenido una precisión bastante aceptable.

Capítulo 57

Sistemas expertos en el ámbito pediátrico

Arturo Peralta, José Ángel Olivas y Eusebio Angulo

57.1. 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 los 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.

57.2. 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 plantee 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 la **Base de Hechos**, que contiene la definición del entorno sobre el que se van a resolver problemas. Hace el papel de "ojos" del Sistema Experto. Además, cuenta con la **Base de Conocimientos**, que contiene el conocimiento del dominio específico convenientemente representado capaz de resolver problemas, y también puede considerar la representación de incertidumbre. Otro componente destacado es el **Motor de Inferencias**, que es el proceso de razonamiento que usa el sistema experto, combina hechos y conocimiento para emitir una conclusión. Por último, también debe tener un **Interfaz de entrada/salida** para comunicarse con los usuarios y/o expertos.

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, reglas prácticas. Es extenso y también redundante e incluye heurística acerca de la heurística y del razonamiento. Es incompleto, impreciso e incierto, (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.

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. Su origen se remonta a 1943 y consta de:

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) 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. Su origen se remonta a 1943 y consta de:

- Conjunto de reglas de producción.
- Memoria de trabajo (Memoria temporal).
- Ciclo de reconocer y actuar.

3) **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 las partes condición de las reglas con las observaciones del mundo con objeto de seleccionar o producir las apropiadas acciones. 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.

4) **Ciclo de reconocimiento y actuación**: Es el procedimiento de control de un sistema de producción. Es un procedimiento de marcha hacia adelante. La memoria de trabajo se inicializa con la descripción del problema. Los modelos guardados en la memoria de trabajo se tratan de superponer en las condiciones de las producciones. Se crea un conjunto "conflicto". Este es 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 hay condiciones en las reglas que cumplan el contenido de los modelos de la memoria de trabajo.

La utilización de los sistemas de producción en SE se remonta a los trabajos de Newell y Simon de la Universidad Carnegie Mellon entre 1960 y 1970. Vieron que los sistemas de producción podían utilizarse para modelar el conocimiento humano y hacer un posterior uso del mismo en la inferencia.

Las principales ventajas de los sistemas de producción en los SE son la separación del conocimiento y del control. Se pueden hacer cambios fáciles de reglas sin cambiar el control y viceversa. Modularidad de las reglas de producción e independencia del lenguaje de programación usado.

57.2.1. Razonamiento

El razonamiento se puede definir 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 el proceso por el que a partir de unos hechos conocidos se obtienen conclusiones

acerca de otros desconocidos (Fleitas, 2017) y (Betanzos A., 2004). La realización de este tipo de procesamiento es llevada a cabo por el denominado **motor de inferencia**.

El razonamiento automático ya se usaba en los 50 en juegos. En 1963 se presentó el sistema *"General Problem Solver"* capaz de hacer inferencias lógicas (Newel y Simon). Aparición del razonamiento heurístico.

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
- *Forward/Backward chaining.*

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 BC el conjunto de reglas aplicables en una situación determinada o estado de la 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.
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 ellos.
3. Las reglas seleccionadas son examinadas para ver si verifican todas sus condiciones. Aquéllas que las verifican constituyen el **"conjunto conflicto"**.
4. De todas aquellas que forman el conjunto conflicto se selecciona una sola y se activa (se dispara). La selección de una regla del conjunto conflicto es la **"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. 57.1 se muestra el algoritmo correspondiente al ciclo de inferencia hacia adelante.

Figura 57.1: Ciclo de razonamiento hacia delante

57.3. 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 en determinadas regiones 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 Triaje) y el CPTAS (Canadian Pediatric Triage and Acuity Scale) 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 de destinar. En este contexto se plantea el desarrollo de un Sistema Experto capaz de ayudar en el proceso de evaluación médica, con el objetivo de facilitar el proceso de triaje.

Como ya se sabe, el primer paso para el desarrollo de un Sistema Experto 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 caso se recopilaron y procesaron un conjunto de datos relativos a 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 Tab. **??**.

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)

Ejemplo de datos de motivos de consulta y Triaje.

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 la aplicación de conocimiento experto, se extrajeron un conjunto de reglas con suficiente calidad. A continuación, se se muestra un ejemplo del 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 Neurológica o 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 Sistema Experto capaz de procesar reglas como las anteriores, realizando una ejecución para tratar de obtener el valor de triaje más posible en caso, por ejemplo, de que el motivo de consulta de una niña sea ginecológico, su edad sea mayor de 12 años, y el tiempo de evolución sea superior a 73 horas.

Es importante señalar que, habitualmente, un sistema experto 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 Sistema Experto: - La Base de Conocimientos del SE contiene 10 reglas.

- Cada regla se modela como una lista de antecedentes y un consecuente.
- La relación entre los consecuentes se modela con el atributo "operador" del siguiente modo: `1=Y=^, 0=o=V, -1=no hay operaciones`

```
r1 <- list(antecedentes = list("Causa_Ginecologica", "Edad_>12a"), consecuente
↪  = list("TiempoEvolucion_>73h"), operador = 0)
r2 <- list(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_Neurológica", "TiempoEvolucion_2-6h"),
↪  consecuente = list("Triaje_4"), operador = 0)
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 Base de Conocimiento con el conjunto de Reglas y la Base de Hechos 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)
```

Se define una función para comprobar la existencia de un número en una lista. Esta función será usada por el motor del Sistema Experto.

```
# 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 Sistema Experto.

El algoritmo ejecuta un bucle en el que, en cada iteración, evalúa las reglas disponibles contenidas en la Base de Conocimiento. Considerando los items de la Base de Hechos, 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 Base de Hechos.Cada regla solo puede dispararse una vez, por lo que se actualiza una lista de reglas risparadas. El algoritmo finaliza cuando el 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]]) {
              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 base de hechos con la situación propuesta (Causa_Ginecologica, Edad_>12a y TiempoEvolucion_>73h), y se ejecuta el motor del Sistema Experto.

```
# 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, su tiempo de espera máximo para recibir atención médica debería ser inferior a 60 minutos. La Tab. 57.3 muestra el proceso realizado por el algoritmo en cada iteración.

IT	Base de Hechos	Conjunto Conflicto	Regla disparada	Consecuente
0	Causa_Ginecologica Edad_>12a TiempoEvolucion_>73h	R1, R2, R4, R10	R1	TiempoEvolucion_>73h
1	Causa_Ginecologica Edad_>12a TiempoEvolucion_>73h Sexo_Mujer"	R2, R4, R10	R2	Sexo_Mujer
2	Causa_Ginecologica Edad_>12a TiempoEvolucion_>73h Sexo_Mujer Causa_Respiratoria	R4, R10	R4	Causa_Respiratoria
3	Causa_Ginecologica Edad_>12a TiempoEvolucion_>73h Sexo_Mujer Causa_Respiratoria Triaje_3	R6, R10	R6	Triaje_3
4	Causa_Ginecologica Edad_>12a TiempoEvolucion_>73h Sexo_Mujer Causa_Respiratoria Triaje_3 Triaje_4	R10	R10	Triaje_4
5	Causa_Ginecologica Edad_>12a TiempoEvolucion_>73h Sexo_Mujer Causa_Respiratoria Triaje_3 Triaje_4	Ø		

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 Base de Hechos 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 Base de Hechos 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 Base de Hechos. 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 Base de Hechos 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 Base de Hechos. 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 Base de Hechos 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 Base de Hechos. 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 Base de Hechos 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 Base de Hechos. 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 la Base de Hechos. Por tanto, el algoritmo finaliza concluyendo como resultado el consecuente de la última regla lanzadas, es decir, `Triaje_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 Sistemas Expertos 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 Sistemas Expertos de un modo rápido y sencillo. Considérese como herramienta de ayuda para situaciones en las que aplicar conocimiento experto resulte clave para la solución de problemas.

Capítulo 58

El NLP y las tendencias en el mundo de la moda

Ambrosio Nguema Ansue

58.1. Introducción

El modelado de temas es el nombre colectivo de las técnicas que ayudan a dividir los textos en temas, por lo que la determinación de los temas relevantes es parte del análisis. Como se trata de una técnica no supervisada, requiere muchas iteraciones. Las técnicas de PNL más novedosas podrían superar a los modelos temáticos en términos de precisión. Sin embargo, una gran ventaja de usar los temas extraídos del texto en una tarea posterior es que la contribución de cada tema (y otras características que tiene disponibles) en el modelo de predicción se analizan y visualizan fácilmente. Esto es útil para cualquier empresa y para cualquiera a quien le esté explicando el modelo predictivo.

58.2. NLP para tendencias de moda en textil

El conjunto `clothes` de datos de reseñas y calificaciones de ropa de comercio electrónico para mujeres contiene 23.486 entradas relacionadas con la edad y la revisión dada por el cliente y sus opiniones sobre la ropa de mujer de varios minoristas.

```
library(CDR)
library(readr)
library(tidytext)
```

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
```

Respecto al **Departamento** con mayor porcentaje de revisiones/calificaciones puede verse que es `Tops` y el que obtiene el menor porcentaje `Jackets`.

```
library(ggplot2)
ggplot(
  data.frame(prop.table(table(clothes$Dept))),
  aes(x = Var1, y = Freq * 100, fill = "blue")
) +
  geom_bar(stat = "identity") +
  xlab("Department Name") +
  ylab("Percentage of Reviews/Ratings (%)") +
  geom_text(aes(label = round(Freq * 100, 2)), vjust = -0.25)
```

Figura 58.1: Percentage of Reviews By Department

En cuanto a las **calificaciones por departamento**, excluida la tendencia, ya que contiene una mezcla de ropa que puede encajar en las otras categorías del Departamento, el análisis se centra en 5 departamentos: `Bottoms`, `Dresses`, `Intimate`, y `Tops`.

```
phisto <- clothes |>
  dplyr::filter(!is.na(Dept), Dept != "Trend") |>
  dplyr::mutate(Dept = factor(Dept)) |>
  group_by(Dept) |>
  dplyr::count(Rating) |>
  dplyr::mutate(perc = n / sum(n))
```

```
phisto |> 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))
```

En cada Departamento, la calificación dominante otorgada es de 5 estrellas. `Jacket` tiene el mayor número de calificaciones de 5 estrellas de todos los departamentos aunque `Jacket` fue el que tuvo la menor cantidad de reseñas.

Si se analizan los **departamentos por edad** se observa que la distribución del número de reseñas por Departamento (i.e. `Tops` con el mayor número de reseñas, `Dresses` con el segundo mayor número de reseñas, etc.) se mantienen dentro de cada uno de los grupos de edad mostrados. Las personas de 30 años dejaron la mayoría de las reseñas, seguidas por las personas de 40 y 50 años. Esto les da a las empresas una idea de quién es el grupo demográfico objetivo y qué tipo de ropa (camisas, vestidos) tienen demanda.

```
ages <- clothes |>
  dplyr::filter(!is.na(Age), !is.na(Dept), Dept != "Trend") |>
  dplyr::select(ID, Age, Dept) |>
  dplyr::mutate(Age_group = ifelse(Age < 30, "18-29", ifelse(Age < 40,
↪  "30-39", ifelse(Age < 50, "40-49", ifelse(Age < 60, "50-59", ifelse(Age <
↪  70, "60-69", ifelse(Age < 80, "70-79", ifelse(Age < 90, "80-89",
↪  "90-99"))))))))

ages <- ages |> mutate(Age_group = factor(Age_group), Dept = factor(Dept,
↪  levels = rev(c("Tops", "Dresses", "Bottoms", "Intimate", "Jackets"))))
```

```
ages |>
  dplyr::filter(Age < 80) |>
  group_by(Age_group) |>
  dplyr::count(Dept) |>
  ggplot(aes(Dept, n, fill = Age_group)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~Age_group, scales = "free") +
  xlab("Department") +
  ylab("Number of Reviews") +
  geom_text(aes(label = n), hjust = 1) +
  scale_y_continuous(expand = c(.1, 0)) +
  coord_flip()
```

Para hacer un **análisis de bigrama**, se eliminan las entradas que no tienen reseñas. Hubo 845 revisiones NA, por lo que 845/23486 * 100 = 3,6 % de calificaciones no se tendrán en cuenta. Además, se combinó el título con la reseña para reunir todas las palabras en una sola sección.

```
clothesr <- clothes |>
  dplyr::filter(!is.na(Review))
notitle <- clothesr |>
  dplyr::filter(is.na(Title)) |>
  dplyr::select(-Title)
wtitle <- clothesr |>
  dplyr::filter(!is.na(Title)) |>
  tidyr::unite(Review, c(Title, Review), sep = " ")
main <- bind_rows(notitle, wtitle)
```

Se ordenan las palabras vacías y se eliminan los dígitos. Se agrupan las palabras de acuerdo con sus calificaciones y se trazaron los 10 bigramas más comunes para cada calificación.

```
bigramming <- function(data) {
  cbigram <- data |> unnest_tokens(bigram, Review, token = "ngrams", n = 2)
  cbigram_sep <- cbigram |>
    tidyr::separate(bigram, c("first", "second"), sep = " ")
  cbigram2 <- cbigram_sep |>
    dplyr::filter(
      !first %in% stop_words$word, !second %in% stop_words$word,
      !stringr::str_detect(first, "\\d"), !stringr::str_detect(second, "\\d")
    ) |>
    tidyr::unite(bigram, c(first, second), sep = " ")
  return(cbigram2)
}

top_bigrams <- bigramming(main) |>
  mutate(Rating = factor(Rating, levels <- c(5:1))) |>
  mutate(bigram = factor(bigram, levels = rev(unique(bigram)))) |>
  group_by(Rating) |>
  dplyr::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") +
  ggtitle("Most Common Bigrams (By Ratings)") +
  coord_flip()
```

No hace falta decir que hay frases positivas para las calificaciones más altas y frases negativas

para las calificaciones más bajas.

Finalmente, se lleva a cabo una visualización mediante una **nube de palabras** para mostrar las palabras compartidas dentro de los bigramas más comunes. Se analizan las reseñas de 5 estrellas y las reseñas de 1 estrella para analizar lo bueno o lo malo de las prendas.

```
onerating <- main |>
  dplyr::filter(Rating == 1)
fiverating <- main |>
  dplyr::filter(Rating == 5)
one <- bigramming(onerating) |>
  dplyr::count(bigram, sort = TRUE)
five <- bigramming(fiverating) |>
  dplyr::count(bigram, sort = TRUE)
```

Nube de palabras para 1 estrella:

```
library(wordcloud2)
wordcloud2(
  one |>
    dplyr::filter(n > 5) |>
    dplyr::mutate(n = sqrt(n)),
  size = .4
)
```

Nube de palabras para 5 estrellas:

```
wordcloud2(five |>
  dplyr::filter(n > 10) |>
  dplyr::mutate(n = sqrt(n)), size = .4)
```

Los **puntos clave** del análisis se resumen en: (i) Las reseñas de 5 estrellas son dominantes en cada departamento y las chaquetas tienen la proporción más alta de todos los departamento. (ii) Los clientes de entre 30 y 40 años dejan la mayoría de las reseñas. El ajuste, la comodidad/calidad del material, la estética de la prenda influyen en la calificación. (iii) Al realizar análisis de datos exploratorio y análisis de bigramas, las empresas pueden concentrarse en lo que funciona y lo que no. Conocer la demografía de los revisores puede informar las decisiones de marketing (por ejemplo, anuncios en línea en los sitios más visitados por personas de 30 y 40 años). Seleccionar artículos que tengan telas flexibles y cómodas puede conducir a una mayor satisfacción del cliente. Un mayor número de críticas positivas se convierte en una forma de publicidad y puede conducir a mayores ventas.

Capítulo 59

Detección de fraude de tarjetas de crédito

Pedro Albarracín

59.1. Introducción

En un informe publicado en diciembre de 2021 por Nilson Report (https://nilsonreport.com/upload/content_promo/NilsonReport_Issue1209.pdf) se informó 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 cometen 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 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 encontremos 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.

59.2. Modelización del fraude en la compra con tarjetas de crérdito

El objetivo que se presenta en este capítulo 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 y contenido en el paquete `CDR` con el nombre de `creditcard`.

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 desbalanceado, lo que añade cierto grado de dificultad al modelo. El dataset un conjunto de 31 variables, de las cuales 28 están identificadas como V1 a V28, una variable "Time" que registra los segundos transcurridos desde esa transacción y la primera, la variable "Amount" que registra el importe de la transacción y la variable dependiente "Clase" que indica con valor 0 que la operación es "no fraudulenta" y con valor 1 las operaciones fraudulentas.

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 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)

# Se obtienen unos datos descriptivos del dataset
library(psych)
describe(creditcard)[1:4, ]
```

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 para X e y
creditcard_X <- creditcard[, -31]
creditcard_y <- creditcard$Class
```

Tratamiento de datos desbalanceados

Uno de los principales problemas a la hora de abordar este tipo de escenarios, es lo que se conoce como "datos desbalanceados". Se deice que un dataset está desbalanceado 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 desbalanceados 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 desbalanceo 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 con los que se obtienen buenos resultados, quizás el más conocido y utilizado sea 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 "Data Mining with R" o DMwR.

Para este caso particular se utilizará una técnica simple de Undersampling basada en el paquete "unbalanced". Unbalanced es un paquete que actualmente no se encuentra disponible en el repositorio de CRAN por lo que para su instalación deberás ejecutar el siguiente código:

```
# install.packages("devtools") // Instalar si no se encuentra entre las
↪  librerías del sistema
library(devtools)
devtools::install_github("dalpozz/unbalanced")
```

Una vez instalado, al igual que todas sus dependencias, se realiza el Undersampling del dataset siguiendo los siguientes pasos:

1.- Convertir la variable dependiente "Class" en factor

```
library(unbalanced)
creditcard$Class <- as.factor(creditcard$Class)
levels(creditcard$Class) <- c("0", "1")
```

2.- A continuación ejecutar la función de Undersampling

```
undersampled_creditcard <- ubBalance(creditcard_X, creditcard$Class, type =
↪  "ubUnder", verbose = TRUE)
undersampled_combined <- cbind(
  undersampled_creditcard$X,
  undersampled_creditcard$Y
)

names(undersampled_combined)[names(undersampled_combined) ==
↪  "undersampled_creditcard$Y"] <- "Class"
levels(undersampled_combined$Class) <- c("Legítima", "Fraude")
#> Proportion of positives after ubUnder : 50% of 984 observations
```

3.- Comprobar el número de casos en el dataset sobre el que se ha ejecutado la función de Undersampling

```
creditcard$Class <- as.factor(creditcard$Class)
levels(creditcard$Class) <- c("0", "1")
```

4.- Realizar la gráfica para visualizar el número de muestras de cada clase después de realizar el Undersampling

```
library(ggplot2)
ggplot(data = undersampled_combined, aes(fill = Class)) +
  geom_bar(aes(x = Class)) +
  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()
  )
```

Figura 59.1: Número de muestras de cada clase después de undersampling (984)

Modelo de Clasificación mediante Regresión Lógística

A continuación se procederá a la construcción de un modelo de regresión logística para una clasificación binaria en relación al fraude en transacciones con tarjeta de crédito a partir de los datos balanceados 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, por un lado, las variables "Time" y "Amount" ya que no van a ser relevantes para el modelo, y cambiar las etiquetas "Legitima" y "Fraude" por 0 y 1 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 seguida del simbolo ~ y un punto (con el punto se hace referencia al resto de variables del dataset)
- 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 los resultados del modelo ejecutar:

```
summary(undersampledModel)
```

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 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)
```

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 O y 1, dependiendo del rango 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 utilizará el siguiente código "ifelse":

```
y_pred <- ifelse(fraud_prob > 0.6, 1, 0)
```

La matriz de confusión

Como último paso del presente estudio se creará la matriz de confusión con el fin de visualizar que tal se ha comportado el algoritmo, es decir, cuántos verdaderos 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 probabilidades resultado del algoritmo. El resultado, como se puede comprobar es que ha evaluado correctamente 184 de las 196 observaciones.

```
confusionMatrix <- table(test[, 29], y_pred)
confusionMatrix
#> y_pred
#>   0  1
#> 0 96 2
#> 1 8 90
```

si hace falta, si no se obvia.

.1. 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] splines   grid      stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>   [1] wordcloud2_0.2.1        readr_2.1.3             MatchIt_4.5.0
↪
#>   [4] tableone_0.13.2         readxl_1.4.1            ggpubr_0.4.0
↪
#>   [7] kableExtra_1.3.4        magrittr_2.0.3          knitr_1.39
↪
#>  [10] coda_0.19-4             spatialreg_1.2-6        spdep_1.2-5
↪
#>  [13] spData_2.2.1            sp_1.5-1                gstat_2.0-9
↪
#>  [16] sf_1.0-9                ggraph_2.0.6            tm_0.7-8
↪
#>  [19] NLP_0.2-1               syuzhet_1.0.6           textdata_0.4.4
↪
#>  [22] tidytext_0.3.4          wordcloud_2.6           RColorBrewer_1.1-3
↪
#>  [25] stopwords_2.3           tokenizers_0.2.3        keras_2.9.0
↪
#>  [28] dtplyr_1.2.1            ca_0.71.1               smacof_2.1-5
↪
#>  [31] colorspace_2.0-3        GPArotation_2022.10-2   psych_2.2.9
↪
#>  [34] FactoMineR_2.6          clValid_0.7             igraph_1.3.4
↪
#>  [37] cluster_2.1.3           dendextend_1.16.0       factoextra_1.0.7
↪
#>  [40] reprtree_0.6            plotrix_3.8-2           tree_1.0-42
↪
#>  [43] devtools_2.4.5          usethis_2.1.6           vip_0.3.2
↪
#>  [46] questionr_0.7.7         epiR_2.0.50             conjoint_1.41
↪
#>  [49] mda_0.5-3               class_7.3-20            quantmod_0.4.20
↪
#>  [52] TTR_0.24.3              xts_0.12.1              zoo_1.8-10
↪
#>  [55] foreign_0.8-82          lubridate_1.9.0         timechange_0.1.1
↪
```

```
#>  [58] forecast_8.18           astsa_1.16              tseries_0.10-51
↪
#>  [61] glmnet_4.1-4            leaps_3.1               ISLR2_1.3-1
↪
#>  [64] mlmRev_1.0-8            lme4_1.1-30             maps_3.4.0
↪
#>  [67] sm_2.2-5.7.1            tidymv_3.3.2            mgcv_1.8-40
↪
#>  [70] nlme_3.1-157            SemiPar_1.0-4.2         Epi_2.47
↪
#>  [73] ResourceSelection_0.3-5 GGally_2.1.2            ggfortify_0.4.14
↪
#>  [76] patchwork_1.1.2         boot_1.3-28             plyr_1.8.8
↪
#>  [79] samplingbook_1.2.4      survey_4.1-1            Matrix_1.5-1
↪
#>  [82] sampling_2.9            pps_1.0                 DescTools_0.99.46
↪
#>  [85] fitdistrplus_1.1-8      survival_3.3-1          Rlab_4.0
↪
#>  [88] rsample_1.1.0           mlbench_2.1-3           gridExtra_2.3
↪
#>  [91] idealista18_0.1.1       corrplot_0.92           ggmosaic_0.3.3
↪
#>  [94] gplots_3.1.3            waffle_0.7.0            ggstatsplot_0.9.4
↪
#>  [97] gapminder_0.3.0         visdat_0.5.3            dlookr_0.6.1
↪
#> [100] mongolite_2.6.2         CDR_0.0.3.0             reshape2_1.4.4
↪
#> [103] dplyr_1.0.10            klaR_1.7-1              MASS_7.3-57
↪
#> [106] e1071_1.7-12            randomForest_4.7-1.1    ipred_0.9-13
↪
#> [109] rpart.plot_3.1.1        rpart_4.1.16            caret_6.0-93
↪
#> [112] lattice_0.20-45         ggplot2_3.4.0
#>
#> loaded via a namespace (and not attached):
#>   [1] styler_1.8.1           remotes_2.4.2          slam_0.1-50
#>   [4] paletteer_1.4.1        labelled_2.10.0        summarytools_1.0.1
#>   [7] expm_0.999-6           vctrs_0.5.1            stats4_4.2.1
#>  [10] utf8_1.2.2             R.oo_1.25.0            withr_2.5.0
#>  [13] gdtools_0.2.4          uuid_1.1-0             lifecycle_1.0.3
#>  [16] emmeans_1.8.1-1        matrixStats_0.62.0     stringr_1.4.1
```

```
#>  [19] cellranger_1.1.0        munsell_0.5.0          ragg_1.2.2
#>  [22] fontawesome_0.4.0       codetools_0.2-18       lmtest_0.9-40
#>  [25] furrr_0.3.1             flashClust_1.01-2      magick_2.7.3
#>  [28] parallelly_1.32.1       fs_1.5.2               ellipse_0.4.3
#>  [31] stringi_1.7.8           polyclip_1.10-0        productplots_0.1.1
#>  [34] pkgconfig_2.0.3         prettyunits_1.1.1      data.table_1.14.6
#>  [37] correlation_0.8.2       estimability_1.4.1     nnls_1.4
#>  [40] httr_1.4.3              flextable_0.8.1        hrbrthemes_0.8.0
#>  [43] modeltools_0.2-23       heplots_1.4-2          mitools_2.4
#>  [46] graphlayouts_0.8.2      haven_2.5.1            parameters_0.18.2
#>  [49] htmltools_0.5.3         miniUI_0.1.1.1         SnowballC_0.7.0
#>  [52] viridisLite_0.4.1       yaml_2.3.5             prodlim_2019.11.13
#>  [55] pillar_1.8.1            later_1.3.0            glue_1.6.2
#>  [58] DBI_1.1.3               foreach_1.5.2          robustbase_0.95-0
#>  [61] gtable_0.3.1            caTools_1.18.2         latticeExtra_0.6-30
#>  [64] rsvg_2.3.2              fastmap_1.1.0          extrafont_0.18
#>  [67] broom_1.0.0             checkmate_2.1.0        promises_1.2.0.1
#>  [70] webshot_0.5.3           FNN_1.1.3.1            rapportools_1.1
#>  [73] textshaping_0.3.6       mnormt_2.1.1           hms_1.1.2
#>  [76] ggforce_0.4.1           askpass_1.1            png_0.1-7
#>  [79] lazyeval_0.2.2          Formula_1.2-4          profvis_0.3.7
#>  [82] crayon_1.5.2            extrafontdb_1.0        svglite_2.1.0
#>  [85] tidyselect_1.2.0        xfun_0.35              performance_0.9.2
#>  [88] kernlab_0.9-31          purrr_0.3.5            pander_0.6.5
#>  [91] etm_1.1.1               spacetime_1.2-8        AlgDesign_1.2.1
#>  [94] rappdirs_0.3.3          xgboost_1.6.0.1        rootSolve_1.8.2.3
#>  [97] jpeg_0.1-9              MatrixModels_0.5-1     pagedown_0.19
#> [100] combinat_0.0-8          ggsignif_0.6.3         R.methodsS3_1.8.2
#> [103] htmlTable_2.4.1         xtable_1.8-4           DT_0.25
#> [106] cachem_1.0.6            gdata_2.18.0.1         statsExpressions_1.3.3
#> [109] abind_1.4-5             systemfonts_1.0.4      mime_0.12
#> [112] weights_1.0.4           gld_2.6.5              ggrepel_0.9.1
#> [115] rstatix_0.7.0           processx_3.7.0         insight_0.18.4
#> [118] numDeriv_2016.8-1.1     tools_4.2.1            cli_3.4.1
#> [121] quadprog_1.5-8          proxy_0.4-27           tibble_3.1.8
#> [124] janeaustenr_1.0.0       future.apply_1.10.0    libcoin_1.0-9
#> [127] assertthat_0.2.1        officer_0.4.4          urca_1.3-3
#> [130] lpSolve_5.6.17          pbapply_1.5-0          s2_1.1.1
#> [133] plotly_4.10.0           R.utils_2.12.2         tweenr_2.0.2
#> [136] BiasedUrn_1.07          effectsize_0.7.0.5     tensorflow_2.9.0
#> [139] zip_2.2.1               bayestestR_0.13.0      tzdb_0.3.0
#> [142] tfruns_1.5.1            ps_1.7.1               lmom_2.9
#> [145] fansi_1.0.3             tidygraph_1.2.2        KernSmooth_2.23-20
#> [148] backports_1.4.1         scatterplot3d_0.3-42   sysfonts_0.8.8
#> [151] interp_1.1-3            farver_2.1.1           shiny_1.7.3
```

```
#> [154] hardhat_1.2.0          whisker_0.4            partykit_1.2-16
#> [157] pROC_1.18.0            viridis_0.6.2          rstudioapi_0.14
#> [160] minqa_1.2.4            iterators_1.0.14       intervals_0.15.2
#> [163] prismatic_1.1.1        shape_1.4.6            gtools_3.9.3
#> [166] inum_1.0-4             rematch2_2.1.2         listenv_0.8.0
#> [169] generics_0.1.3         base64enc_0.1-3        pkgbuild_1.3.1
#> [172] ModelMetrics_1.2.2.2   BayesFactor_0.9.12-4.4 pryr_0.1.5
#> [175] R.cache_0.16.0         timeDate_4021.106      evaluate_0.16
#> [178] memoise_2.0.1          candisc_0.8-6          doParallel_1.0.17
#> [181] httpuv_1.6.6           Rttf2pt1_1.3.10        Rcpp_1.0.9
#> [184] polynom_1.4-1          openssl_2.0.4          classInt_0.4-8
#> [187] diptest_0.76-0         dbscan_1.1-10          pkgload_1.3.0
#> [190] cmprsk_2.2-11          jsonlite_1.8.3         Hmisc_4.7-2
#> [193] fracdiff_1.5-1         digest_0.6.30          showtextdb_3.0
#> [196] bookdown_0.28          rprojroot_2.0.3        LearnBayes_2.15.1
#> [199] bitops_1.0-7           here_1.0.1             rmarkdown_2.14
#> [202] globals_0.16.2         compiler_4.2.1         nnet_7.3-17
#> [205] reticulate_1.26        carData_3.0-5          tcltk_4.2.1
#> [208] rlang_1.0.6            urlchecker_1.0.1       nloptr_2.0.3
#> [211] prabclus_2.3-2         reactable_0.3.0        wk_0.7.0
#> [214] sessioninfo_1.2.2      fpc_2.2-9              lava_1.7.0
#> [217] multcompView_0.1-8     rvest_1.0.2            recipes_1.0.3
#> [220] future_1.29.0          mvtnorm_1.1-3          htmlwidgets_1.5.4
#> [223] forcats_0.5.2          labeling_0.4.2         callr_3.7.1
#> [226] flexmix_2.3-18         datawizard_0.6.1       curl_4.3.2
#> [229] parallel_4.2.1         highr_0.9              DEoptimR_1.0-11
#> [232] scales_1.2.1           showtext_0.9-5         deldir_1.0-6
#> [235] Exact_3.1              mice_3.15.0            car_3.1-0
#> [238] zeallot_0.1.0          tidyr_1.2.1            ellipsis_0.3.2
#> [241] xml2_1.3.3             gower_1.0.0            reshape_0.8.9
#> [244] R6_2.5.1               mclust_5.4.10          units_0.8-0
```

Bibliografía

Aas, K., Jullum, M., and Løland, A. (2021). Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502.

Abedjan, Z., Golab, L., and Naumann, F. (2015). Profiling relational data: a survey. *The VLDB Journal*, 24(4):557–581.

Abraham, R., Schneider, J., and vom Brocke, J. (2019). Data governance: A conceptual framework, structured review, and research agenda. *International Journal of Information Management*, 49:424–438.

acens.com (2014). White paper: Bbdd nosql. Report.

Al-Ruithe, M., Benkhelifa, E., and Hameed, K. (2019). A systematic literature review of data governance and cloud data governance. *Personal and Ubiquitous Computing*, 23(5-6):839–859.

Allaire, J. (2022). *quarto: R Interface to 'Quarto' Markdown Publishing System*. R package version 1.2.

Almudevar, A. (2021). *Theory of Statistical Inference*. Texts in Statistical Science. Chapman & Hall/CRC.

Amat Rodrigo, J. (2017). *Clustering y heatmaps: aprendizaje no supervisado*.

Amazon Web Services (2018). ¿qué es nosql? Report.

Anderberg, M. R. (1973). *Cluster analysis for applications: probability and mathematical statistics*. Academic press, New Yoor, USA.

Anscombe, F. J. (1973). Graphs in statistical analysis. *American Statistician*.

Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Studies in Operational Regional Science. Springer Netherlands.

Anselin, L. (1995). Local indicators of spatial association—lisa. *Geographical Analysis*, 27(2):93–115. Citation Key: anselinLocalIndicatorsSpatial1995.

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.*

Arnab, R. (2017). *Survey Sampling Theory and Applications.* Elsevier.

Astigarraga, J. and Cruz-Alonso, V. (2022). ¡se puede entender cómo funcionan git y github! *Ecosistemas*, 31(1):2332.

Azevedo, A. and Santos, M. F. (2008). Kdd, semma and crisp-dm: a parallel overview. *IADS-DM.*

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.

Bakka, H., Rue, H., Fuglstad, G.-A., Riebler, A., Bolin, D., Illian, J., Krainski, E., Simpson, D., and Lindgren, F. (2018). Spatial modeling with r-inla: A review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(6):e1443.

Balakrishnan, N., Koutras, M. V., and Politis, K. G. (2019). *Introduction to Probability: Models and Applications.* Wiley Series in Probability and Statistics. John Wiley & Sons.

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.

Batini, C., Scannapieco, M., et al. (2016). Data and information quality. *Cham, Switzerland: Springer International Publishing.*

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.

Betanzos A., G. B. (2004). *Ingeniería del conocimiento: Aspectos metodológicos.* Pearson Education.

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.

Blais, B. (2020). *Statistical Inference for everyone.* Save the broccoli publishing.

Blaug, M. (1980). *La metodología de la economía o cómo explican los economistas.* Alianza Editorial.

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 Rupretch, Göttingen, Germany.

Boehmke, B. and Greenwell, B. (2019). *Hands-on machine learning with R.* Chapman and Hall/CRC.

Boehmke, B. y Greenwell, B. (2020). *Hands-On Machine Learning with R.* Chapman and Hall/CRC. Chapman and Hall.

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.

Brachman, R. J. and Anand, T. (1994). The process of knowledge discovery in databases: A first sketch. In *KDD workshop*, volume 3, pages 1–12.

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 editio.

Brockwell, P. J. and Davis, R. A. (2016). *Introduction to Time Series and Forecasting.* Springer Texts in Statistics. Springer International Publishing, Switzerland.

Brous, P., Herder, P., and Janssen, M. (2016). Governing Asset Management Data Infrastructures. *Complex Adaptive Systems Los Angeles, CA November 2-4, 2016*, 95:303–310.

Brownlee, J. (2020). *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python.* Machine Learning Mastery.

Bunge, M. (2004). *La investigación científica. Su estrategia y su filosofía.* SIGLO XXI Editores, Buenos Aires.

Bunge, M. (2018). *La ciencia: su método y su filosofía*, volume 1. Laetoli.

Burkov, A. (2019). *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada.

Caballero, I., Gualo, F., Rodríguez, M., and Piattini, M. (2022a). Br4dq: A methodology for grouping business rules for data quality evaluation. *Information Systems*, 109:102058.

Caballero, I., Gómez-Carretero, A., Gualo, F., Merino, J., Rivas, B., and Piattini, M. (2017). Calidad de datos. *Ra-Ma*.

Caballero, I., Piattini, M., and Gualo, F. (2022b). Marco metodológico para la creación, implantación y mantenimiento de Sistemas de Gobierno de Datos. In Goñi Sarriguren, A., editor, *JISBD2022*. SISTEDES.

Caballero, I., Piattini, M., and Rodríguez, M. (2020). Modelo alarcos de mejora de datos v3.0: Un modelo de referencia de procesos basado en estándares internacionales abiertos para la gestión de los datos, gestión de la calidad de los datos y el gobierno de los datos. Technical report, DQTeam, UCLM, AQCLab.

Cancelo, J. R. (1997). *Proyecto Docente e Investigador. Concurso de acceso al cuerpo docente de Catedráticos de Universidad.* Universidad de Castilla-La Mancha.

Carrasco-Oberto, G. I. (2020). *Cluster no jerárquicos versus cart y biplot.* PhD thesis, Universidad de Salamanca.

Carruthers, C. and Jackson, P. (2020). *The chief data officer's playbook.* Facet Publishing.

Casella, G. and Berger, R. L. (2007). *Statistical inference, 2nd Ed.* Cengage Learning.

Chacon, S. (2009). *Pro Git.* Apress.

Chalmers, A. F., Villate, J. A. P., Máñez, P. L., and Sedeño, E. P. (2000). ¿qué es esa cosa llamada ciencia?

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., et al. (2000a). Crisp-dm 1.0: Step-by-step data mining guide. *SPSS inc*, 9(13):1–73.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., et al. (2000b). Crisp-dm 1.0: Step-by-step data mining guide. *SPSS inc*, 9(13):1–73.

Chasco Yrigoyen, C. (2006). Análisis estadístico de datos geográficos en geomarketing: El programa GeoDa. *Distribución y consumo*, 178:34.

Chatfield, C. and Collins, A. (1980). *Introduction to multivariate analysis.* Chapman&Hall/CRC.

Chaudhuri, A. and Stenger, H. (2005). *Survey Sampling. Theory and Methods, 2nd Ed.* STATISTICS: a series of TEXTBOOKS and MONOGRAPHS. Chapman & Hall/CRC.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

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.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Council, E. (2020). The Data Capability Assessment Model (DCAM) Framework v2.2 Overview.

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. 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.

Cruz Rivera, S., Liu, X., Chan, A.-W., Denniston, A. K., Calvert, M. J., Ashrafian, H., Beam, A. L., Collins, G. S., Darzi, A., Deeks, J. J., ElZarrad, M. K., Espinoza, C., Esteva, A., Faes, L., Ferrante di Ruffano, L., Fletcher, J., Golub, R., Harvey, H., Haug, C., Holmes, C., Jonas, A., Keane, P. A., Kelly, C. J., Lee, A. Y., Lee, C. S., Manna, E., Matcham, J., McCradden, M., Moher, D., Monteiro, J., Mulrow, C., Oakden-Rayner, L., Paltoo, D., Panico, M. B., Price, G., Rowley, S., Savage, R., Sarkar, R., Vollmer, S. J., and Yau, C. (2020). Guidelines for clinical trial protocols for interventions involving artificial intelligence: the spirit-ai extension. *The Lancet Digital Health*, 2(10):e549–e560.

Cryer, J. D. and Chan, K.-S. (2010). *Time Series Analysis with Applitacions in R.* Springer texts in Statistics. Springer, Iowa, USA.

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.*

DAMA (2017). *DAMA-DMBOK: data management body of knowledge.* Technics Publications, LLC.

Davenport, T. and Harris, J. (2017). *Competing on analytics: Updated, with a new introduction: The new science of winning.* Harvard Business Press.

Davenport, T. H. and Patil, D. (2012). Data scientist. *Harvard business review*, 90(5):70–76.

Davies, T. M. and Baddeley, A. (2018). Fast computation of spatially adaptive kernel estimates. *Statistics and Computing*, 28(4):937–956.

De Boor, C. (2001). *A practical guide to splines*. Applied Mathematical Sciences. Springer-Verlag, New York.

de Finetti, B. (2017). *Theory of Probability: A Critical Introductory Treatment*. Wiley Series in Probability and Statistics. Wiley.

De la Fuente, S. (2011). *Análisis Factorial*. Madrid, España.

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.

Díez, J. A. and Moulines, C. U. (2008). *Fundamentos de Filosof?a de la Ciencia*. Editorial Ariel, Barcelona.

Efron, B. and Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pages 54–75.

Elhorst, J. P. (2010). Applied Spatial Econometrics: Raising the Bar. *Spatial Economic Analysis*, 5(1):9–28.

Engels, B. (2019). Data Governance as the Enabler of the Data Economy. *Intereconomics*, 54(4):216–222.

Eryurek, E., Gilad, U., Laksmanan, V., Kibunguchy-Grant, A., and Ashdown, J. (2021). *Data Governance: The Definitive Guide*. O'Reilly Media, Inc.

Faraway, J. J. (2002). *Practical regression and ANOVA using R.*, volume 168. University of Bath Bath.

Fay, C., Rochette, S., Guyader, V., and Girard, C. (2021). *Engineering Production-Grade Shiny Apps*. Chapman and Hall/CRC.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. (1996). Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88.

Firth, J. (1957). A synopsis of linguistic theory, 1930–1955. *En 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. (2019). *Cuentapalabras*. 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 Grananda, 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.

García-Alsina, M. (2017). *Big data: gestión y explotación de grandes volúmenes de datos*. UOC.

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 Rhing, 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.

Gómez-Rubio, V. (2020). *Bayesian inference with INLA*. CRC Press.

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.

Gualo, F., Rodríguez, M., Verdugo, J., Caballero, I., and Piattini, M. (2021). Data quality certification using ISO/IEC 25012: Industrial experiences. *J. Syst. Softw.*, 176:110938.

Guerry, A.-M. (1833). *Essai Sur La Statistique Morale de La France*. Crochard.

Gómez García, J. L. and Conesa i Caralt, J. (2015). *Introducción al big data.* Oberta Barcelona, UOC Publishing.

Hajek, A. and Hitchcock, C. (2016). *The Oxford Handbook of Probability and Philosophy.* Oxford University Press.

Hamilton (1994). *Time Series Analysis.* Statistics. Princenton University Press, Princeton, NJ, USA.

Harman, H. H. (1976). *Modern Factor Analyis (Third Edition Revised.* Chicago, USA.

Harrison, T., F. Luna-Reyes, L., Pardo, T., De Paula, N., Najafabadi, M., and Palmer, J. (2019). The Data Firehose and AI in Government: Why Data Management is a Key to Value and Ethics. In *Proceedings of the 20th Annual International Conference on Digital Government Research*, dg.o 2019, pages 171–176, New York, NY, USA. Association for Computing Machinery. event-place: Dubai, United Arab Emirates.

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.

Hecht, R. and Jablonski, S. (2011). Nosql evaluation: A use case oriented survey. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 336–341. IEEE.

Hempel, C. (2005). *La explicación científica: Estudios sobre la filosofía de la ciencia.* Ediciones Paidós, Barcelona.

Henderson, C. (1953). Estimation of variance and covariance components. *Biometrics*, 9:226–252.

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.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice.* OTexts.

igraph.org (2022). *Network Analysis and Visualization.*

Ihaka, R. and Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314.

IIC, I. d. I. d. C. (2016). Las 7 v del big data: Características más importantes. Report.

Illian, J., Penttinen, A., Stoyan, H., and Stoyan, D. (2008). *Statistical Analysis and Modelling of Spatial Point Patterns*, volume 70. John Wiley & Sons.

Ilyas, I. F. and Chu, X. (2019). *Data cleaning*. Morgan & Claypool.

ISACA (2019). COBIT | Control Objectives for Information Technologies.

ISO (2016). ISO 8000-61:2016.

ISO (2017). ISO/IEC 38505-1:2017 Information technology — Governance of IT — Governance of data — Part 1: Application of ISO/IEC 38500 to the governance of data.

ISO (2018). ISO 8000-62:2018.

ISO (2018). ISO/IEC TR 38505-2:2018 Information technology — Governance of IT — Governance of data — Part 2: Implications of ISO/IEC 38505-1 for data management.

ISO/IEC (2008a). Iso/iec 25012: Software engineering-software product quality requierements and evaluation (square) - data quality model. Report.

ISO/IEC (2008b). Iso/iec 25024: Software engineering-software product quality requierements and evaluation (square) - data quality model. Report.

ISO/IEC (2011). So/iec 25040 -systems and software engineering — systems and software quality requirements and evaluation (square) — evaluation process. Report.

ISO/IEC (2015). Iso 8000-8: Data quality — part 8: Information and data quality: Concepts and measuring. Report.

Jackson, P. and Carruthers, C. (2019). *Data Driven Business Transformation: How to Disrupt, Innovate and Stay Ahead of the Competition*. 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/.

Janssen, M., Brous, P., Estevez, E., Barbosa, L. S., and Janowski, T. (2020). Data governance: Organizing data for trustworthy artificial intelligence. *Government Information Quarterly*, 37(3):101493.

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.

Johnson, N. L., Kemp, A. W., and Kotz, S. (2008). *Univariate Discrete Distributions, 3e Set.* Wiley Series in Probability and Statistics. Wi.

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.

Kalyvas, J. R. and Overly, M. R. (2014). *Big data: a business and legal guide.* CRC Press.

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.

Khatri, V. and Brown, C. V. (2010). Designing Data Governance. *Commun. ACM*, 53(1):148–152. Place: New York, NY, USA Publisher: Association for Computing Machinery.

Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466.

Kim, J.-H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational statistics & data analysis*, 53(11):3735–3745.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27(2):97–111.

Kohonen, T. (1995). *Learning vector quantization.* Springer.

Krainski, E., Gómez-Rubio, V., Bakka, H., Lenzi, A., Castro-Camilo, D., Simpson, D., Lindgren, F., and Rue, H. (2018). *Advanced spatial modeling with stochastic partial differential equations using R and INLA.* Chapman and Hall/CRC.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of statistical software*, 28:1–26.

Kuhn, M. (2019). CRAN Task View: Reproducible Research. R Task View.

Kuhn, M., Johnson, K., et al. (2013). *Applied predictive modeling*, volume 26. Springer.

Ladley, J. (2019). *Data governance: How to design, deploy, and sustain an effective data governance program.* Academic Press.

Lancaster, T. (2004). *An introduction to modern Bayesian econometrics.* Blackwell Oxford.

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.

Lillie, T. and Eybers, S. (2019). *Identifying the constructs and agile capabilities of data governance and data management: A review of the literature*, volume 933 of *Communications in Computer and Information Science*. Pages: 326.

Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

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.

Loshin, D. (2002). Rule-based data quality. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 614–616.

Loshin, D. (2011). Master data management and data quality bt-the practitioner's guide to data quality improvement. In *MK Series on Business Intelligence*, pages 327–350. Morgan Kaufmann.

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, D. (2012). Análisis de las posibilidades de uso de big data en las organizaciones. *Universidad de Cantabria, Santander, España*.

López-Gónzalez, 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.

Mahanti, R. (2019). *Data quality: dimensions, measurement, strategy, management, and governance*. Quality Press.

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.

Martins, T. G., Simpson, D., Lindgren, F., and Rue, H. (2013). Bayesian computing with inla: new features. *Computational Statistics & Data Analysis*, 67:68–83.

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*.

Matejka, J. and Fitzmaurice, G. (2017). Same Stats, Different Graphs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1290–1294, New York, NY, USA. ACM.

Matheron, G. (1962). *Traité de Geostatistique Apliquée. vol I*. Éditions Technip, Paris, France.

Matloff, N. and Zhang, W. (2022). A novel regularization approach to fair ml. *arXiv preprint arXiv:2208.06557*.

McNicholas, P. D. (2016). Model-based clustering. *Journal of Classification*, 33(3):331–373.

Mecca, M., Young, R., and Halcomb, J. (2014). *Data Management Maturity (DMM) Model*. CMMI Institute, first edition.

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.

Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307.

Møller, J. and Waagepetersen, R. (2003). *Statistical Inference and Simulation for Spatial Point Processes.* CRC Press.

Montero, J. M. (1997). *Proyecto Docente e Investigador. Concurso de acceso al cuerpo docente de Catedráticos de Universidad.* Universidad de Castilla-La Mancha.

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.* Wiley series in probability and statistics. Wiley, Chichester, West Sussex.

Montero, J. M. and Larraz, B. (2008). *Introducción a la Geoestadística Lineal.* Metodología y Análisis de Datos en Ciencias Sociales. Netbiblo, A Coruña, España.

Montero Lorenzo, J. (2007). Estadística descriptiva, editorial thomson-paraninfo.

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.

Moraga, P. (2019). *Geospatial health data: Modeling and visualization with R-INLA and shiny.* CRC Press.

Morin, D. J. (2016). *Probability: For the Enthusiastic Beginner.* CreateSpace Independent Publishing Platform.

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.

OECD (2019). *The Path to Becoming a Data-Driven Public Sector*. OECD Digital Government Studies. OECD Publishing, Paris.

Okabe, A. and Sugihara, K. (2012). *Spatial Analysis Along Networks: Statistical and Computational Methods*. John Wiley & Sons.

Olmeda, M. V. and Ibánez, J. C. (2022). *Manual de ética aplicada en inteligencia artificial*. ANAYA MULTIMEDIA.

O'neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway books.

Osimo, D., Mureddu, F., Peristeras, V., Cioffi, A., Moise, C., and van, C. (2020). Data Strategies, Policies and Agenda. page 17.

Osorio-Sanabria, M. A., Amaya-Fernández, F., and González-Zabala, M. P. (2020). Developing a model to readiness assessment of open government data in public institutions in Colombia. pages 334–340.

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 Scientarum Imperialis Petropolitane 8.

Pearson, K. (1904). On the theory of contingency and its relation to association and normal correlation. In Department of Aplied 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.

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.

Piattini, M., Marcos, E., Calero, C., and Vela, B. (2006). Tecnología y diseño de bases de datos. *Editorial Ra-Ma*.

Pizarro, M., Hernangómez, D., and Fernández-Avilés, G. (2021). *climaemet: Climate AEMET Tools*.

Plotkin, D. (2020). *Data stewardship: An actionable guide to effective data management and data governance*. Academic Press.

Price, R. and Shanks, G. (2004). A semiotic information quality framework. In *Proceedings of the International Conference on Decision Support Systems DSS04*, pages 658–672.

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.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In *Machine learning*, pages 463–482. Springer.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Rakshit, S., Davies, T., Moradi, M., McSwiggan, G., Nair, G., Mateu, J., and Baddeley, A. (2019). 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.

Redman, T. C. (2016). *Getting in front on data: who does what.* Technics Publications.

Revelle, W. (2022). *psych: Procedures for Psychological, Psychometric, and Personality Research.* Evanston, Illinois. R package version 2.2.5.

Rey, D., Arbués, P., López, F., and Páez, A. (2022). A geo-referenced micro-data set of real estate listings for spain's three largest cities. Working paper.

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.*

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.

Ross, S. (2012). *A First Course in Probability 9th ed.* Prentice Hall, Upper Saddle River, NJ 07458.

Rubenfa (2014). Mongodb: empezando por el principio. insertando datos. Report, Genbeta Dev.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the royal statistical society: Series b (statistical methodology)*, 71(2):319–392.

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.

Saeys, Y., Inza, I., and Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517.

Sakia, R. M. (1992). The box-cox transformation technique: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(2):169–178.

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.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer, New York. ISBN 978-0-387-75968-5.

SAS Institute Inc. (2017). Big data. what it is and why it matters.

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.

Shafique, U. and Qaiser, H. (2014). A comparative study of data mining process models (kdd, crisp-dm and semma). *International Journal of Innovation and Scientific Research*, 12(1):217–222.

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.

Singer, J. and Willet, J. (2003). *Applied Longitudinal Data Analysis*. Oxford University Press.

Snijers, T. (2003). Fixed and random effects. 2:664–665.

Soares, S. (2010). *The IBM data governance unified process: driving business value with IBM software and best practices*. MC Press, LLC.

Soares, S. (2015). *The chief data officer handbook for data governance.* Mc Press New York.

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.*

Strong, D., Lee, Y., and Wang, R. (1997a). 10 potholes in the road to information quality. *Computer*, 30(8):38–46.

Strong, D. M., Lee, Y. W., and Wang, R. Y. (1997b). Data quality in context. *Communications of the ACM*, 40(5):103–110.

Strozzi, C. (1998). Nosql-a relational database management system. Report.

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.

Tierney, N. (2017). visdat visualising whole data frames. *Journal of Open Source Software*, 2(16):355.

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.

Treder, M. (2020). *The Chief Data Officer Management Handbook.* Apress, Berkeley, CA.

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.

Tukey, J. W. (1962). The future of data analysis. *The annals of mathematical statistics*, 33(1):1–67.

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.

Venables, W. and Ripley, B. (2002). *Modern applied statistics with S.* Statistics and computing. 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.

Wang, R. Y. (1998). A product perspective on total data quality management. *Communications of the ACM*, 41(2):58–65.

Webb, G. I. (2011). Filtered-top-k association discovery. *WIREs Data Mining and Knowledge Discovery*, 1(3):183–192.

Weber, K., Otto, B., and Österle, H. (2009). One Size Does Not Fit All—A Contingency Approach to Data Governance. *J. Data and Information Quality*, 1(1). Place: New York, NY, USA Publisher: Association for Computing Machinery.

Wickham, H. (2015). *Advanced R.* Chapman & Hall/CRC The R Series. CRC Press.

Wickham, H. (2016). *ggplot2.* Use R! Springer International Publishing, Cham, second edi edition.

Wickham, H. and Grolemund, G. (2016). *R for Data Science.* O'Reilly Media.

Wickham, H. and Wickham, M. H. (2017). Package tidyverse. *Easily Install and Load the 'Tidyverse.*

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.

Wood, S. N. (2006). *Generalized Additive Models - An introduction with R.* Texts in Statistical Science. Chapman & Hall.

Wu, C. and Thompson, M. E. (2020). *Sampling Theory and Practice*. ICSA Book Series in Statistics. Springer.

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

Deviance, 254
'fluidPage()', 703

agregación, 270
ajuste automático, 376, 387–389, 399, 413, 421, 434, 439
algoritmo bayesiano, 403
algoritmos de árbol, 366, 371
ANOVA, 220
análisis de colocaciones, 585
Análisis de Componentes Principales, 477
análisis de datos, 28
análisis de redes, 598
análisis de sentimientos, 586, 587, 592
análisis de textos, 583
análisis exploratorio de datos, 127
Análisis Factorial, 489
aplicaciones web interactivas, 701
aprendizaje ensamblado, 411, 427
arista, 367
AUC, 167, 239
Autocorrelación espacial, 653, 657
autovalor, 479

bagging, 411–416, 427, 430
base, 253
batch, 548
Big Data, 39
binning, 160
bondad del ajuste, 221
boosting, 411, 427, 430, 436
bootstrap, 411, 412, 416, 418
bootstrapping, 166
botón circular, 706

caja negra, 391
ciencia, 27
    empírica, 30
    formal, 30
ciencia de datos, 28
clasificación, 237, 535, 539–541, 547, 549, 560, 562, 567, 575, 579
clasificación binaria, 385
clasificación multiclase, 386
codificación, 160
coeficientes de regresión, 220
comercio, 761
complejidad, 370, 385, 417, 418, 436
Componentes principales, 290
componentes reactivos, 711
construcción, 765
contraste de Breusch-Pagan, 226
Contraste de esfericidad de Bartlett, 494
contraste de Shapiro-Wilk, 226
control deslizante, 706
coordenadas, 620
Coordinate reference system, 620
corpus, 584
correlación, 153
covariables, 235
COVID-19, 759
CRS, 620
CRS geográficos, 622
CRS proyectados, 623
cuarteto de Anscombe, 127
Curva ROC, 167
curva ROC, 239

Daniel G. Krige, 649
data augmentation, 569, 570
datos
    integración, 122
    no balanceados, 163
    preprocesamiento, 115

problemas de calidad, 116
visualización, 119
datos de patrones de puntos, 633
datos espaciales, 619
datos geoestadísticos, 633
datos geográficos, 619
datos jerárquicos, 269
datos lattice, 633
datos longitudinales, 271
datos raster, 625
datos reticulares, 633
datos vector, 624
deep learning, 533–536, 538, 548, 569
Dependencia espacial, 653
desagregación, 270
descenso del gradiente, 535, 547, 564
detección, 536
detección de emociones, 587, 593
Deviance, 290
diagrama de dispersión, 223
Diagrama de Moran, 657
distancia, 396, 446
Chebychev, 448
de Mahalanobis, 448
entre centroides, 453
euclídea, 446
euclídea al cuadrado, 446
Manhattan, 447
media, 453
Minkowski, 448
Distancia absoluta o city-block, 511
Distancia de Gower, 512
distancia de Gower, 397
Distancia de Jaccard, 512
Distancia de Sokal-Michener, 512
distancia euclidea, 396
Distancia euclídea, 511
distancia intragrupos, 456
Distancia Mahalanobis, 511
distancia manhattan, 396
distancias angulares, 622
distribución, 763
distribución normal, 219
Dropout, 568

Early Stopping, 568
ecuaciones de Henderson, 273
edad, 759
efectos aditivos, 235
efectos aleatorios, 271
efectos fijos, 271
efectos multiplicativos, 235
elastic net, 304
Empirical semivariogram, 646
encadenamiento completo, 452
encadenamiento simple, 452
entropía, 167, 369, 416, 418
error del modelo, 220
escala, 395
Escalamiento Multidimensional, 509
Escalamiento métrico, 514
Escalamiento no-métrico, 518
especificidad, 239
estadística, 28
Estadística Descriptiva, 127
estandarización, 297
estilometría, 586
estimación de coeficientes, 237
exactitud, 121, 168
expecificidad, 168
extreme gradient boosting, 436

factores, 235
feature engineering, 159
feature selection, 151
Fisher-Jenks, 628
frequency, 782
frontera de decisión, 383
función de enlace, 234, 235
función de perdida, 386

Gini, 167
gradient boosting, 417, 428, 431, 436, 437
grados de libertad efectivos, 254
Gráfico de Sedimentación, 483

hard margin, 386
hiperparametro, 396
hiperparámetro, 376, 387, 389, 413, 417, 418, 430, 431, 436
hiperparámetros, 167
hiperplano, 386
hoja, 367

huella lingüística, 586

I de Morna, 659
importancia, 391, 392, 414
impureza de Gini, 368, 418
imputación, 120
indepencia condicional, 404
industria, 765
Inferencia Estadística, 127
informática, 28
inteligencia artificial, 533
intercambiabilidad, 271
interfaz de usuario, 702
Internet of Things, 39
Interpolation, 649
Interpretación de modelos espaciales, 665

John Snow, 619

k-fold cross-validation, 300
k-means, 785
k-vecinos, 395, 397, 399
kernel, 387, 392

lematización, 586
lexicón, 588
lm, 222

machine learning, 391, 533–535
mapa, 627
mapa de calor, 761
margen, 383, 385, 386
matemática, 28
matriz de documentos, 587, 595
matriz de similaridad, 511
mayores de 45 años, 767
Medida de adecuación muestral de Kaiser, Meyer y Oklin (KMO), 494
metodología, 33
    CRISP-DM, 34
    KDD, 34
    SEMMA, 34
Metodología CRISP-DM, 782
minería de opinión, 587
minería de textos, 583
missing, 119
    aleatorios, 119
    informativos, 119
modelado de temas, 586
modelo aditivo, 251
modelo de regresión, 219
modelo de regresión lineal múltiple, 219
modelo lineal generalizado, GLM, 219, 234
modelo no lineal, 219
Modelo RFM, 781
modelo sparse, 302
Modelos econométricos espaciales, 660
modelos mixtos lineales, 272
monetary, 782
MSE, 167
muestra, 127
muestreo
    aleatorio simple, 162
    estratificado, 162
multicolinealidad, 221
Máxima Verosimilitud, 648
Máxima Verosimilitud Restringida, 648
método, 27
    científico, 27
método de la mediana, 454
Método de los factores principales, 497
Método de Máxima Verosimilitud, 499
método de máxima verosimilitud, 234
método de vinculación entre grupos, 456
método de Ward, 455
método Elbow, 786
Método RFE, 154
Métodos basados en Máxima Verosimilitud, 648
mínimos cuadrados, 220, 223
Mínimos cuadrados generalizados, 648
Mínimos cuadrados ordinarios, 648
Mínimos cuadrados ponderados, 648

n-gramas, 585, 596
Naive Bayes, 404, 405
neurona, 538, 539, 541, 545, 547, 560, 564
nodo, 367
nodo raiz, 367
nubes de palabras, 587, 592, 595
número de árboles, 417, 430

observaciones independientes, 220

odd ratio, OR, 238
outlier, 118

P-splines, 252
p-valor, 220, 237
padding, 564
palabras vacías, 584, 590
Paquete Caret, 150
parada temprana, 370
paro de larga duración, 764
paro registrado, 759
particionado de datos, 161
partición, 368, 416, 418
parámetros de suavizado, 252
penalización shrinkage, 296
perceptrón, 535, 538–540, 545, 547, 560, 564
perceptrón multicapa, 545, 560, 579
perfil de los clientes, 781
PLN-procesamiento del lenguaje natural, 583
población, 127
poda, 371
pooling, 566, 567
precisión, 121, 168
predicción, 237
predicción del modelo, 234
predicción del modelo lineal, 222
preprocesamiento, 573
profundidad del árbol, 370, 418, 431
programación, 28
punto de decisión, 367

R, 39
R cuadrado, 221
rama, 367
random forest, 415–418, 427, 428, 430, 431
ranking de percentiles, 782
reactividad, 711
recency, 782
reconocimiento, 536
recuento, 233
redes neuronales artificiales, 538, 560–562
redes neuronales convolucionales, 535, 561, 579
Regionalización, 641
regionalización, 649
regresión lasso, 302
regresión logística, 236
regresión ridge, 296
regularización, 436
remuestreo, 165
residuos, 221
residuos estandarizados, 274
residuos parciales, 260
residuos studentizados, 274
restauración, 761
riesgo relativo, RR, 238
Rotaciones oblicuas, 500
Rotaciones ortogonales, 500
Rotación ORTOMAX, 501
Rotación QUARTIMAX, 501
Rotación VARIMAX, 501
RSME, 167
ruido estadístico, 121

S-Stress, 513
saber, 27
Scale parameter, 645
sector, 759
segmentación, 536
segmentar, 781
selección
    tipo embedded, 155
    tipo filtro, 152
    tipo wrapper, 154
selección de variables, 151, 289
selección stepwise, 293
selección stepwise backward, 293
selección stepwise forward, 293
Semivariograma
    exponencial, 644
    gausiano, 644
    nugget, 644
semivariograma, 639
    esférico, 643
    isotrópico, 643
sensibilidad, 168, 239
servidor, 702
sexo, 759
Shape parameter, 645
Shrinkage, 290
shrinkage, 296
significativo, 220, 237
Sistema de referencia de coordenadas, 620

sobreajuste, 370, 412, 427, 430, 436, 437, 568
soft margin, 386
spline, 257
splines, 649
stemming, 585
stopwords, 584, 597
Stress de Kruskal, 513
stride, 565
suavizadores, 251
subconjunto testing, 295
subconjunto training, 295
SVM, 385, 392

tabsetPanel(), 705
tasa de aprendizaje, 430
teorema de Bayes, 403–405
tibble, 759
tiempo de búsqueda de empleo, 759
token, 585
tokenización, 585
truco del kernel, 386
Tukey, 28
turismo, 761
Técnicas de agrupación, 443
término independiente, 220

valiación cruzada, 165
variable explicativa, 219
variable respuesta, 219, 233
vecino más cercano, 452
vecino más lejano, 452
visualización, 127

weight decay, 569

árbol de clasificación, 411, 417, 418
árbol de decisión, 411, 412, 414, 416, 418, 427, 428, 431, 436
árbol de regresión, 411, 417, 418
árboles de clásificación, 365, 374
árboles de decisión, 365, 367, 371
árboles de regresión, 365, 378