Herramientas del BIG DATA y analytics

Introducción a la detección de anomalías empleando R

Autores:
Julio César Alonso
Estefenía Serrano I.

Universidad Icesi
Editorial Universidad Icesi

Introducción a la detección de anomalías empleando R

Julio César Alonso C[1] Estefenía Serrano I[2]

2025-10-19

[1]Departamento de Economía - Facultad de Negocios y Economía - Universidad Icesi, jcalonso@icesi.edu.co
[2]Departamento de Economía - Facultad de Negocios y Economía - Universidad Icesi, eserrano@icesi.edu.co

Julio César Alonso C. - Estefania Serrano I.

Colección «Herramientas del Big Data y Analytics», vol. 8

Cali. Universidad Icesi, 2025.

184 páginas.

Incluye referencias bibliográficas.

ISBN: 978-628-7814-18-9 (eBook).

DOI: https://doi.org/10.18046/EUI/bda.h.8

Palabras Clave: 1. R | 2. Analítica | 3. Detección de anomalías | 4. Outliers | 5. Big Data Analytics

Clasificación Dewey: 545 ddc 21

Rector: Esteban Piedrahita Uribe

Secretaria General: Olga Patricia Ramírez Restrepo

Director Académico: José Hernando Bahamón

Coordinador editorial: Adolfo A. Abadía

Corrección de estilo: Paola Vargas Heredia

Diseño de portada: Sandra Moreno

Fotos tomadas por: Julio César Alonso

Editorial Universidad Icesi

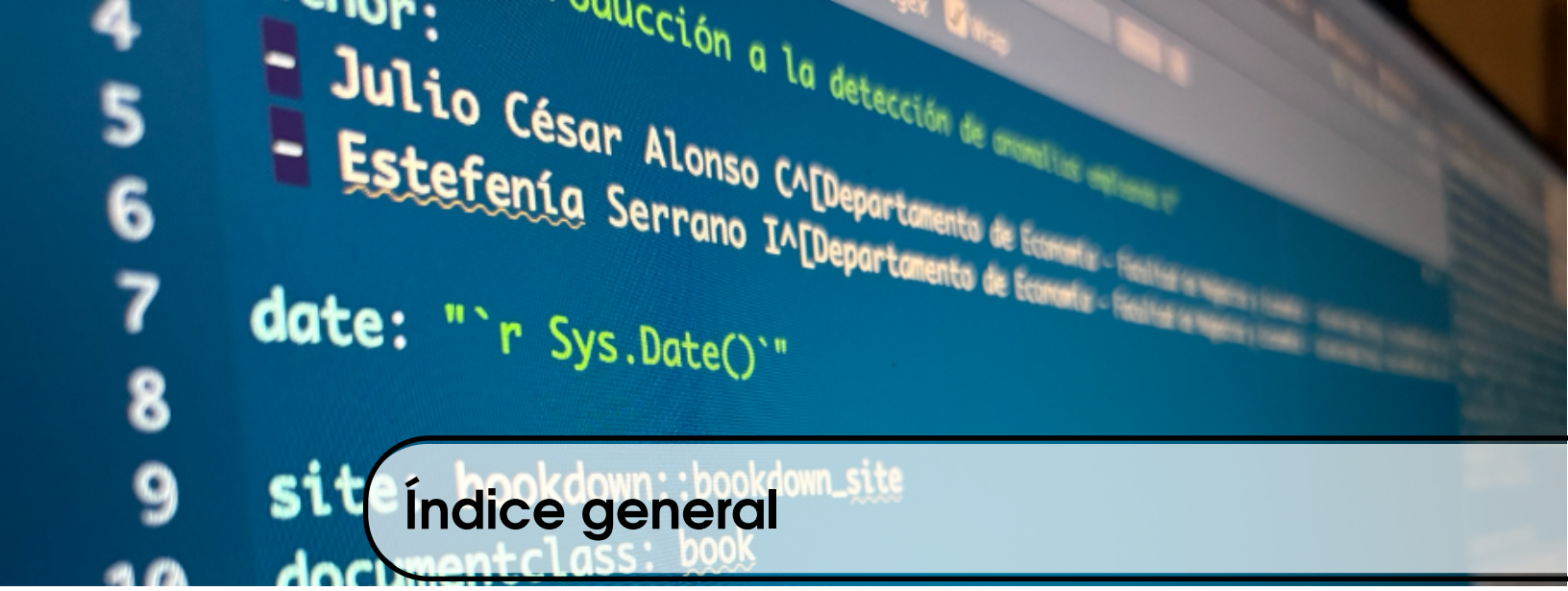
Calle 18 No. 122-135 (Pance), Cali – Colombia

Teléfono: +57 (2) 555 2334 | E-mail: editorial@icesi.edu.co

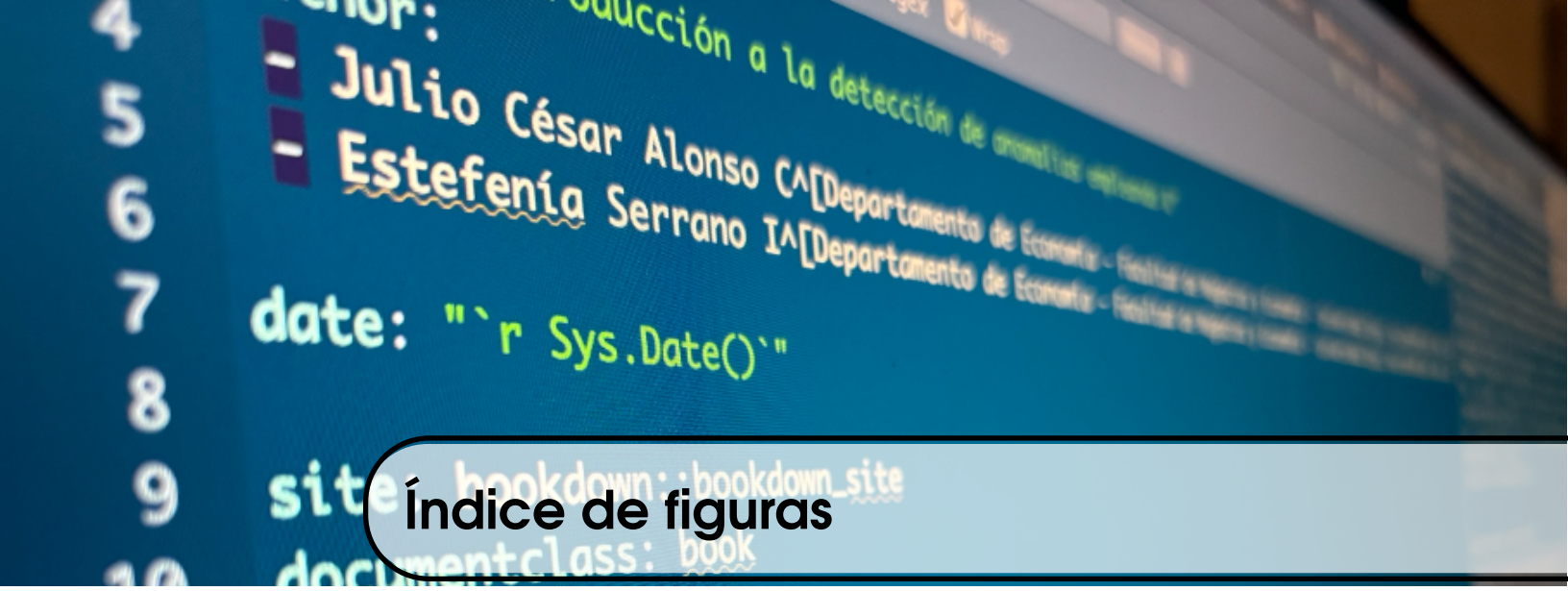
http://www.icesi.edu.co/editorial

Publicado en Colombia – *Published in Colombia*

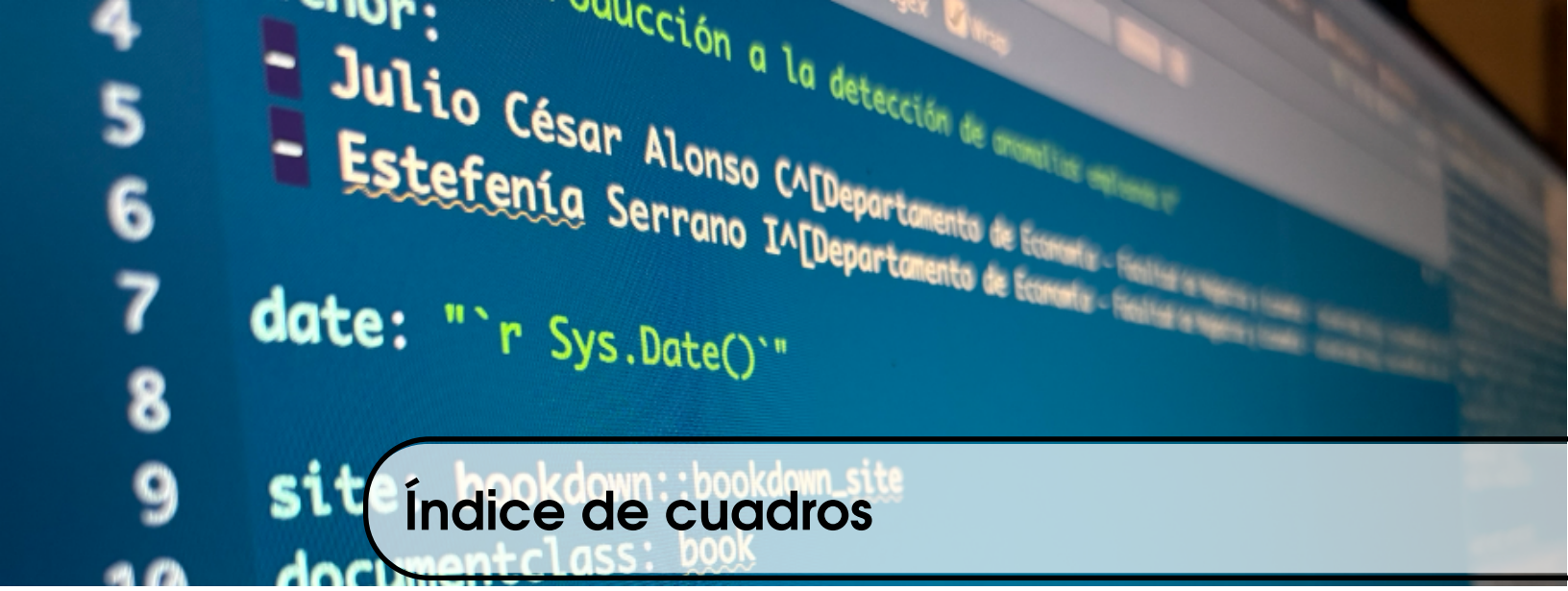
La publicación de este libro se aprobó luego de superar un proceso de evaluación doble ciego.



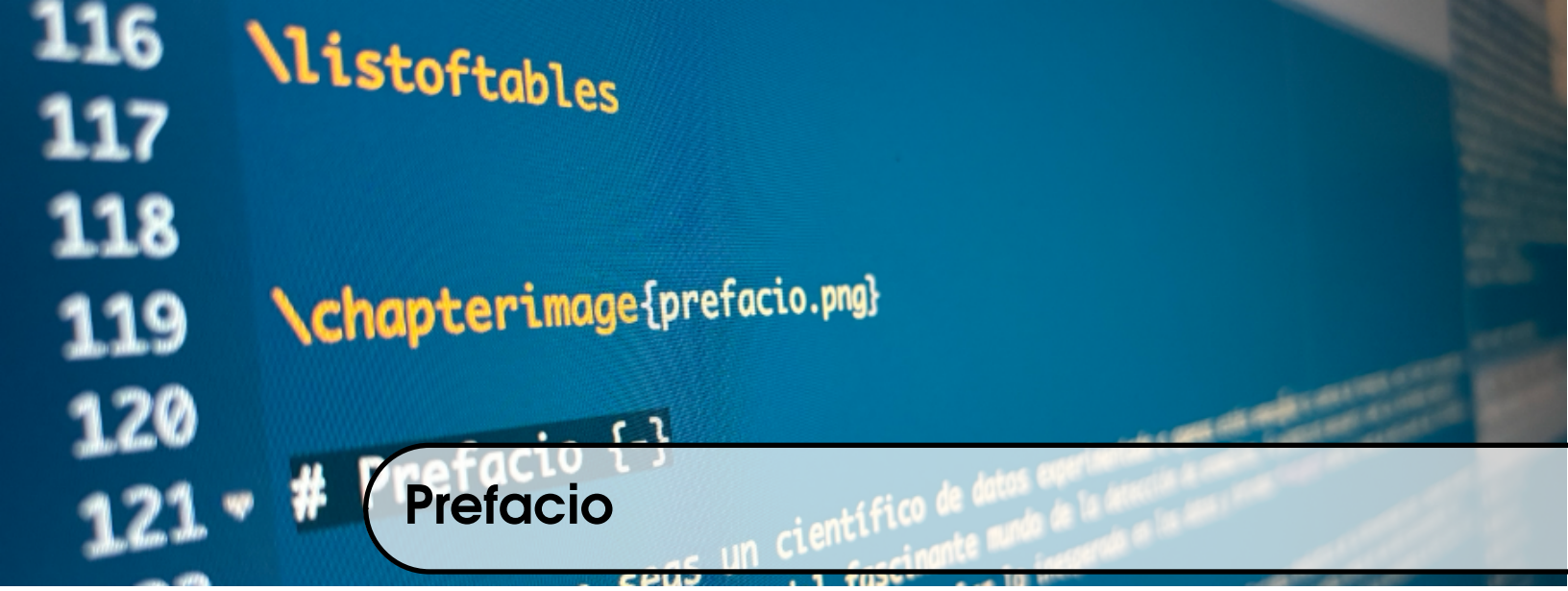
Índice general



Índice de figuras



Índice de cuadros



Prefacio

Ya sea que seas un científico de datos experimentado o apenas estés empezando tu camino de formación, este libro te ayudará en empezar el estudio del fascinante mundo de la detección de anomalías. Te permitirá descubrir cómo los diferentes modelos y técnicas pueden ayudarte a revelar lo inesperado en los datos y brindar *insights* para la toma de decisiones más informadas y estratégicas basadas en datos.

Este libro surge de las notas de clase del curso Introducción al Business Analytics de la Universidad Icesi. Después de emplear por varios años parte del material que conforma esta obra, decidimos convertir estas notas en una obra autocontenida. Los contenidos de los capítulos y su arquitectura, son producto de los comentarios valiosos de los estudiantes de este curso y los investigadores del Cienfi, con quienes estamos agradecidos.

Este libro presenta una introducción a los modelos estadísticos y de aprendizaje de máquina que permiten realizar la tarea de identificación de anomalías en R (R Core Team, 2025). La detección de anomalías es una tarea útil en diferentes áreas del mundo de negocios para encontrar comportamientos atípicos. En este libro explicaremos esto con mayor detalle. Así mismo, en cada capítulo presentaremos aplicaciones en R (R Core Team, 2025) que te permitirán practicar lo aprendido. La discusión de los diferentes capítulos está dirigida a personas que están empezando su formación de científico de datos. Es decir, a personas que están construyendo su caja de herramientas que le permitan realizar tareas típicas de un científico de datos.

Con esta obra completamos una colección de 6 libros que introducen 6 de las tareas rutinarias de un científico de datos:

- *Visualizar* (Alonso, 2022)
- *Clusterizar* (*Agrupar*) (Alonso et al., 2025)
- *Clasificar* (Alonso y Hoyos, 2025)
- *Encontrar reglas de asociación* (Alonso y Arboleda, 2025)
- *Estimar regresiones* (Alonso, 2024)

Esperamos que esta obra pueda ayudarte en tu camino de formación en aprender técnicas del *business analytics* y cómo aplicarlas empleando R.

Esta obra recoge nuestra experiencia trabajando con R y detección de anomalías para resolver problemas con datos desde el Cienfi (Centro de Investigación en Economía y Finanzas) de la Universidad Icesi. En el Cienfi, empleamos R para la transformación de datos en conclusiones que faciliten la toma de decisiones en organizaciones privadas y públicas. Toda esta experiencia la queremos plasmar en esta obra para asegurar que nuevas generaciones de profesionales continúen fortaleciendo la comunidad de R alrededor del mundo.

Este libro supone un uso intermedio de R. Si crees que necesitas algún refuerzo en R, te recomendamos tres libros. Alonso y Ocampo (2022) presenta una breve introducción para iniciar a usar R. Ese primer libro discute cómo instalar R y RStudio y paquetes, cómo cargar diferentes bases de datos y cómo realizar operaciones aritméticas y lógicas con objetos. En ese libro también se discuten las clases esenciales de objetos sencillos y compuestos. No dudes en consultar ese primer libro si aún no has iniciado tu camino por el universo de R.

El segundo libro de la serie (Alonso, 2022) presenta una breve introducción al paquete para *dplyr* (Wickham et al., 2021) que permite manipular objetos que contengan datos. En ese libro se discute cómo filtrar observaciones, crear nuevas variables y combinar objetos con datos. Es recomendable tener un conocimiento de ese paquete antes de leer esta obra. Consulta ese segundo libro si aún no has tenido alguna experiencia manipulando objetos con datos con *dplyr*.

Finalmente, te recomendamos leer (Alonso y Largo, 2023) en el que se presenta una introducción a la creación de visualizaciones con el paquete *ggplot2* (Wickham, 2016). En esta obra emplearemos visualizaciones construidas con este paquete. Así este libro asume un manejo intermedio de R, y los paquetes *dplyr* y *ggplot2*.

Por otro lado, es deseable que antes de leer este libro tengas conocimientos a nivel introductorio de modelos que permitan realizar la tarea de clasificación y la tarea de clústering. Una introducción a los modelos de clasificación en R la puedes encontrar en Alonso y Hoyos (2025) y una introducción a la construcción de conglomerados en R la puedes consultar en Alonso et al. (2025).

Este libro está organizado en tres grandes partes. La primera parte que corresponde al Capítulo 1 y presenta una introducción a la tarea de detección de anomalías y discute qué es una anomalía y los tipos de anomalía. La segunda parte, que va del Capítulo 2 al 6, se concentra en los métodos de origen en la estadística para detectar anomalías. En el Capítulo 2 se discute el uso de estadísticas descriptivas para encontrar anomalías en una sola variable (anomalías univariadas). Así mismo, en ese capítulo discutimos la labor del análisis exploratorio de los datos, también conocido por su sigla en inglés **EDA** (*Exploratory Data Analysis*), y cómo la detección de anomalías es parte de esa labor. En el Capítulo 3, a diferencia del Capítulo 2, se discuten pruebas estadísticas para encontrar *outliers* univariados. En este capítulo también introducimos el proceso conocido como Extracción, Transformación y Carga (**ETL** por el término en inglés *Extract-Transform-Load*) en el que también las técnicas para encontrar anomalías

es importante.

El Capítulo 4 presenta pruebas estadísticas para encontrar observaciones anómalas multivariadas; es decir, observaciones que son anomalías no solamente teniendo en cuenta una sola variable. En el Capítulo 5 en el que discutimos cómo emplear la técnica de Componentes Principales (**PCA** por su sigla en inglés) para detectar anomalías multivariadas. Finalmente, la segunda parte del libro la cierra el Capítulo 6, capítulo en el que hacemos un pequeño alto en el camino para discutir qué es un fraude, cuáles son los tipos de fraude y las propiedades deseables de un buen modelo para detectar fraude. Así mismo, ahí presentamos la Ley de Benford que es una herramienta convencional y potente para detectar fraudes.

La tercera parte del libro presenta métodos de aprendizaje de máquina para la detección de anomalías multivariadas. En el Capítulo 7 se presenta una breve discusión al aprendizaje de máquina (*Machine Learning*) y su relación con el campo de la Inteligencia Artifical. En el Capítulo 8 se discuten métricas para detectar anomalías de origen en el aprendizaje de máquina como la distancia **kNN** y **LOF**. En el Capítulo 9 se discute el uso del modelo de clúatering **DBSCAN** para detectar anomalías. Finalmente, el Capítulo 10 presenta los modelos de **Isolation Tree** y **Isolation Forest** para detectar anomalías.

El libro concluye con una cuarta parte muy corta compuesta por el Capítulo 11 que presenta un cierre a esta obra y discute otros modelos para la detección de anomalías que no fueron cubiertas en este libro.

¡Esperamos encuentres esta obra útil y la compartas con otros futuros usuarios interesados! Si tienes alguna sugerencia del libro o corrección, no dudes en escribirnos. Esta es una obra en constante construcción.

Parte I

Definición y tipos de anomalías

1 . Introducción

En los últimos años se ha acelerado la transformación del mundo de los negocios por la cada vez más grande disponibilidad de mayores volúmenes de información y capacidad de cómputo que permiten emplear modelos de analítica para responder preguntas de negocio y así generar valor a las organizaciones (Alonso, 2024). Emplear herramientas de *business analytics* y *Big Data* permite tanto optimizar los procesos actuales de la organización como generar características diferenciadoras para cada organización.

Los datos se han convertido en un recurso estratégico para las organizaciones, y el *business analytics* es la herramienta con la que las organizaciones pueden monetizar ese recurso (Alonso, 2024). El *business analytics* es el proceso científico de transformar datos en *insights* con el propósito de tomar mejores decisiones (Alonso, 2024).

En el mundo del *business analytics* nos enfrentamos a preguntas de negocios que implican identificar lo inesperado, lo excepcional, lo que se desvía de lo habitual. Este desafío es fundamental en muchas áreas de las organizaciones: por ejemplo, en la planta de producción se desea detectar el producto anómalo y en el área contable se desea encontrar fraudes. La identificación de lo inesperado, lo excepcional, lo que se desvía de lo habitual, se conoce en el mundo del *business analytics* como la **tarea de detección de anomalías**.

En este libro, exploraremos en profundidad algunos de los modelos que se emplean en la detección de anomalías y su implementación en R (R Core Team, 2025). Como lo discutiremos, encontrar lo inesperado, lo excepcional, lo que se desvía de lo habitual dependerá en parte del contexto del negocio y del tipo de datos de los que dispongamos. Así mismo, las herramientas (modelos) que se emplean para encontrar estas anomalías son muy diferentes y tienen diferente naturaleza.

Antes de entrar en el detalle de las técnicas, estudiaremos en este capítulo qué es la tarea de detección de anomalías, qué tipo de analítica se puede desarrollar con esta tarea, qué se entiende por anomalía y los tipos de anomalías que podemos encontrar. Esto permitirá entender de manera más fácil las herramientas que tenemos a nuestra

disposición para la detección de anomalías y cuándo se debería emplear cada una de ellas.

1.1 Tareas de analítica y tipos de analítica

El proceso científico de transformar datos en *insights* para la toma de decisiones parte de una buena pregunta de negocio. A partir de los datos disponibles, los científicos de datos deberán escoger la mejor herramienta para responder a la pregunta de negocio planteada. Las respuestas a las preguntas de negocio implican diferentes tareas que se pueden clasificar en una de las siguientes categorías[1]:

- *Clasificar*[2].
- *Estimar regresiones*[3].
- **Detectar anomalías**.
- *Clusterizar* (*Agrupar*)[4].
- *Encontrar reglas de asociación* (*o buscar coocurrencia de productos*)[5].
- *Pronosticar*.
- *Resumir*.
- *Visualizar*[6].

La tarea de *detección de anomalías* tiene como finalidad encontrar al individuo (observación) que se comporta de manera diferente a los demás. En otras palabras, la tarea de detección de anomalías se emplea para encontrar aquel caso (observación) que no sigue el patrón de las otras observaciones.

Por ejemplo, en algunas situaciones se deseará determinar si el patrón de compra de un sujeto se desvía significativamente de lo normal (norma establecida). Otro ejemplo podría ser en el área de producción, donde se desea identificar productos defectuosos que no cumplen con los estándares de calidad establecidos. En el área de finanzas y contabilidad, la detección de anomalías puede emplearse para identificar transacciones financieras inusuales que podrían indicar fraude o errores contables. En el área de recursos humanos, la detección de anomalías puede ayudar a identificar patrones de comportamiento inusuales entre los empleados que podrían indicar problemas de desempeño o mal uso de los recursos de la organización. En el área de logística, la detección de anomalías puede emplearse para identificar patrones inusuales en la cadena de suministro que podrían indicar problemas en la entrega o distribución de productos. En la Figura 1.1 se presenta una representación gráfica de esta tarea.

[1] Ver Alonso y Serrano (2025) para una discusión completa de qué es el *****business analytics******, la importancia de la pregunta de negocio y los elementos que componen una buena pregunta de negocio.

[2] Para una discusión detallada de esta tarea y cómo implementarla en R se puede consultar Alonso y Hoyos (2025)

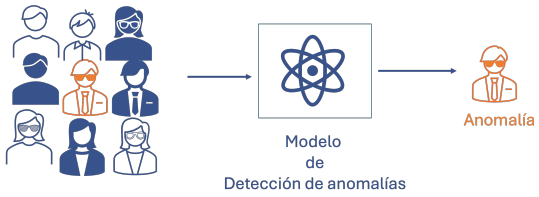
[3] Para una discusión detallada del modelo clásico de regresión y cómo implementarlo en R se puede consultar Alonso (2024).

[4] Para una discusión detallada de esta tarea y cómo implementarla en R se puede consultar Alonso et al. (2025).

[5] Para una discusión detallada de esta tarea y cómo implementarla en R, se puede consultar Alonso y Arboleda (2025).

[6] Para una discusión detallada de esta tarea y cómo implementarla en R se puede consular Alonso (2022).

Figura 1.1. Tarea de detección de anomalías



Fuente: elaboración propia.

Las preguntas de negocio que se pueden responder con la tarea de detección de anomalías pueden ser muy variadas. Por ejemplo, en el área de *mercadeo* pueden surgir preguntas como:

- ¿Cuáles son los patrones de comportamiento de los clientes en línea que podrían indicar actividad fraudulenta?
- ¿Qué campañas publicitarias han generado un aumento inusual en el tráfico del sitio web?
- ¿Existen segmentos de clientes que están mostrando un comportamiento de compra atípico?

En el área de *ventas* las preguntas pueden ser del tipo:

- ¿Qué transacciones de ventas son significativamente mayores o menores que lo habitual y requieren una revisión más detallada?
- ¿Qué regiones geográficas están experimentando un aumento inusual en las ventas?
- ¿Existen productos que están experimentando un cambio repentino en la demanda?

En el área de *finanzas y contabilidad* de una organización pueden surgir preguntas de negocio que implican la tarea de detección de anomalías como las siguientes:

- ¿Qué transacciones financieras son anómalas y podrían indicar fraude?
- ¿Cuáles son los patrones de gastos inusuales que podrían indicar problemas en la gestión financiera?
- ¿Cuáles son las transacciones contables que no cumplen con las normativas o políticas internas?
- ¿Qué cuentas muestran cambios inusuales en los saldos?
- ¿Existen patrones de facturación que podrían indicar actividades fraudulentas?

En el área de *recursos humanos* también pueden surgir preguntas como:

- ¿Cuáles son los patrones de comportamiento de los empleados que podrían indicar un mal uso de los recursos de la organización?
- ¿Existen discrepancias inusuales en los registros de asistencia o tiempo de trabajo?
- ¿Qué tendencias de rotación de personal son atípicas y podrían requerir intervención?

En el área de *producción* las preguntas pueden ser como las siguientes:

- ¿Qué patrones de producción son anómalos y podrían indicar problemas de calidad o mantenimiento?
- ¿Existen discrepancias inusuales en los niveles de inventario?
- ¿Qué máquinas o equipos están experimentando un aumento inusual en el tiempo de inactividad?

En el área de *logística* las preguntas podrían ser como las siguientes:

- ¿Cuáles son los patrones de distribución que son atípicos y podrían indicar problemas en la cadena de suministro?
- ¿Existen retrasos inusuales en la entrega de productos a los clientes?
- ¿Qué rutas de transporte están experimentando un aumento inusual en los costos?

En general, las preguntas pueden ser muy variadas y dependen del área funcional y de la industria a la que pertenece la organización. La gama de preguntas de negocio que se pueden responder con la tarea de detección de anomalías es muy grande.

Por otro lado, es importante anotar que los datos que se emplean para realizar la detección de anomalías pueden corresponder a muestras de corte transversal[7] o series de tiempo[8]. Las muestras pueden o no contener una variable con anomalías previamente marcadas o no. Es decir, en algunas situaciones podemos tener datos en los cuales ya se rotuló cuáles observaciones son anomalías y cuáles no. En este caso, la tarea de detección de anomalías corresponderá a modelos de aprendizaje supervisado[9], en especial modelos de clasificación. Para este tipo de modelos, la muestra contiene la variable objetivo (es anomalía o no) que se encuentra etiquetada[10]. La misión del modelo o algoritmo es aprender de esos datos cuál es el patrón que permitiría predecir la categoría (anomalía o no) de un individuo dadas las características de este. En este libro no estudiaremos los modelos de detección de anomalías de aprendizaje supervisado. Si quieres estudiar estos modelos, puedes estudiar los modelos de clasificación que pueden servir también para detectar anomalías. Para estudiar los modelos de clasificación te recomendamos consultar a Alonso y Hoyos (2025).

En otras oportunidades, la muestra con la que se cuenta no tiene rotuladas las observaciones que se consideran anomalías. En esta situación, la misión del algoritmo que empleemos es detectar cuáles observaciones son anómalas. En este caso la tarea de detección de anomalías implicará emplear modelos de aprendizaje no supervisado, pues la muestra no contiene una variable objetivo que se encuentra etiquetada con la

[7] Es decir, a muchos individuos se les observa sus características y la variable objetivo (si aplica) en el mismo período.

[8] En otras palabras, se observa el mismo individuo en diferentes periodos de tiempo, regularmente espaciados.

[9] En el campo del aprendizaje de máquina se distinguen dos tipos de aprendizaje: supervisado y no supervisado. En el aprendizaje no supervisado, los datos no contienen etiquetas o la "respuesta correcta". El aprendizaje no supervisado busca descubrir patrones o estructuras ocultas en los datos. En el Capítulo 7 se presenta una breve introducción a estos conceptos. En el Capítulo 11 de Alonso y Serrano (2025) se presenta una introducción mas extensa a estos conceptos.

[10] Es decir, con la respuesta correcta ya conocida.

pertenencia o no de una observación a un grupo[11]. La misión del modelo o algoritmo es aprender de esos datos cuál es el patrón que permitiría determinar los individuos (dadas las características de estos) que son anómalos.

En otras palabras, en el caso de la tarea de detección de anomalías, dependiendo de la muestra disponible y de la pregunta de negocio, se pueden emplear modelos de aprendizaje supervisado o no supervisado. En este libro nos concentraremos en modelos de aprendizaje no supervisado para detectar anomalías.

Regresando a las tareas del *business analytics*, una manera más común de clasificar los ejercicios de analítica es según su propósito (Alonso y Serrano, 2025). En ese caso se tienen cuatro tipos de analítica (Ver Figura 1.2):

- **Analítica Descriptiva**: Esta analítica se enfoca en resumir y visualizar los datos para obtener información sobre lo que ha sucedido en el pasado. Ayuda a comprender patrones y tendencias.
- **Analítica Diagnóstica**: Esta analítica busca entender por qué algo ha sucedido. Examina los datos para identificar las causas raíz de los problemas o éxitos pasados.
- **Analítica Predictiva**: Esta analítica utiliza modelos estadísticos y de aprendizaje de máquina para hacer pronósticos y predecir eventos futuros.
- **Analítica Prescriptiva**: Esta analítica se centra en recomendar acciones y soluciones óptimas para lograr un objetivo.

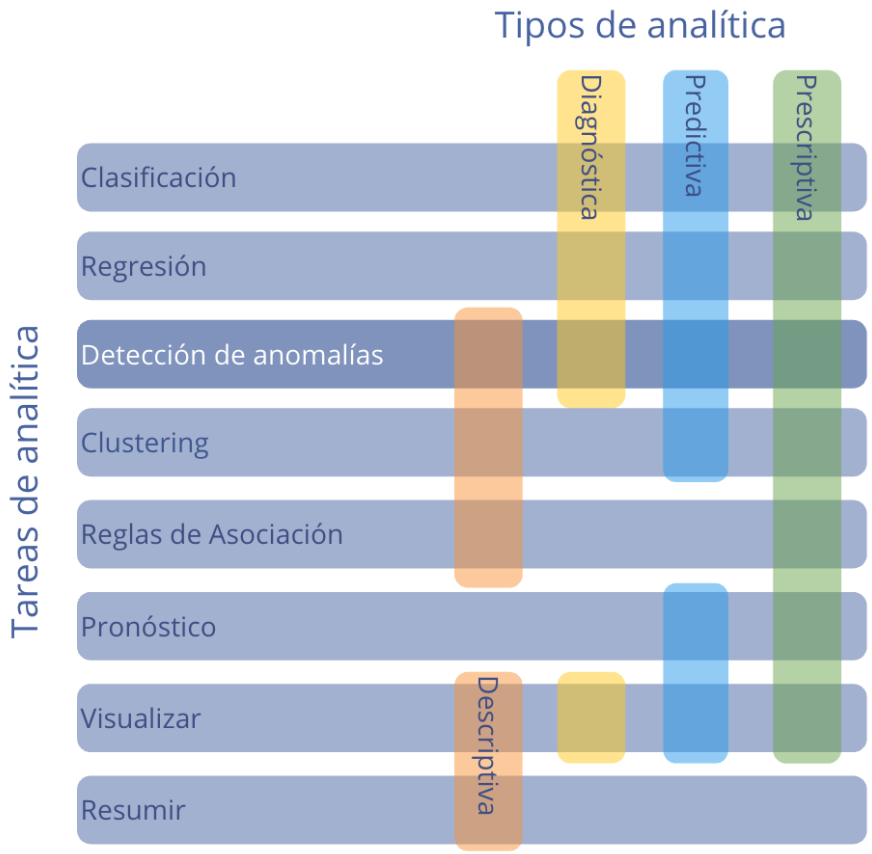
Figura 1.2. Material multimedia: tipos de analítica

Fuente: elaboración propia. https://rb.gy/s7efwr

Estos cuatro tipos de analítica engloban las 8 tareas del *business analytics*. La tarea de detección de anomalías permite realizar los cuatro tipos de analítica (Ver Figura 1.3).

[11] Es decir, la respuesta "correcta" de ser anomalía no es conocida.

Figura 1.3. Relación entre las tareas de analítica y los tipos de analítica



Fuente: tomado de Alonso y Serrano (2025).

La **analítica descriptiva** responde a la pregunta **¿qué está pasando en el negocio?**[12]. La detección de anomalías permite comprender qué está pasando en un momento determinado. Por ejemplo, puede identificar transacciones financieras inusuales, patrones de comportamiento de clientes atípicos o fluctuaciones en los datos de producción que requieren atención. Esta información descriptiva es fundamental para comprender la situación actual de la organización. De hecho, algunos de los métodos o algoritmos que se emplean en esta tarea se consideran como métodos exploratorios de datos, pues nos permiten entender características de un conjunto de datos. En el Capítulo 2 estudiaremos métodos que se pueden incluir en la etapa de exploración de datos para realizar cualquier tipo de tarea de analítica.

La **analítica diagnóstica** responde a la pregunta **¿por qué está pasando algo?** En el contexto de las preguntas de detección de anomalías, dependiendo de los modelos y datos que se empleen, se podrá determinar por qué un individuo es considerado una anomalía. Esto podrá ocurrir en especial cuando se puedan emplear modelos de aprendizaje supervisado y modelos de clasificación como el Logit. La detección de anomalías se puede emplear para comprender por qué ciertos eventos inusuales están ocurriendo. Por ejemplo, podría identificar que las ventas anormalmente bajas de un periodo se deben a un cambio en la estrategia de marketing. Este nivel de análisis ayuda a identificar las causas raíz de los problemas o tendencias inusuales.

La **analítica predictiva** busca responder a la pregunta **¿qué es posible que ocurra?** Los modelos de detección de anomalías también pueden emplearse para predecir futuros eventos inusuales. Por ejemplo, podría predecir posibles fraudes basándose en patrones de comportamiento pasados o anticipar problemas de calidad en la producción. Esto permite a las organizaciones tomar medidas preventivas antes de que ocurran eventos no deseados.

La **analítica prescriptiva** busca responder a la pregunta **¿qué necesito hacer?** La detección de anomalías puede emplearse para recomendar acciones específicas para abordar eventos inusuales. Por ejemplo, podría recomendar cambios en las políticas de seguridad informática para evitar brechas de seguridad o ajustes en la cadena de suministro para mitigar problemas de logística. Esto ayuda a las organizaciones a tomar decisiones informadas y proactivas para mejorar su desempeño.

Hasta aquí hemos hablado de la tarea de analítica, el tipo de preguntas que puede responder y el tipo de análisis que se quiere realizar, pero aún no hemos definido qué es una anomalía. En la siguiente sección nos concentraremos en esto.

1.2 Anomalías: definición y tipos

Una anomalía es algo que difiere "significativamente" de lo que se considera común o "normal"[13]. En otras palabras, la tarea implica buscar una desviación, un error,

[12] Para ver una introducción rápida al tipo de datos que se emplean en el *business analytics* ver el video disponible en el siguiente enlace: https://youtu.be/2OxY2UTI_Bs.

[13] En este contexto el término "normal" no se refiere a la distribución normal sino a la acepción más común y coloquial del término. Es decir, por normal queremos decir habitual u ordinario.

un evento inusual, una excepción o patrones de comportamiento diferentes. Una anomalía corresponde a un evento raro (observación) que no encaja en el patrón y, por tanto, parece sospechoso. Una anomalía puede definirse como una observación o un conjunto de ellas que no parece seguir el mismo patrón que el resto de los datos en una o varias de sus características. Es decir, ¡es como la tarea de encontrar una aguja en un pajar!

Anomalías y outliers

El término *outlier* (o valor atípico en español) nace de la estadística y corresponde a una observación que difiere significativamente de las otras observaciones (Grubbs, 1969). Inicialmente el interés de los estadísticos y usuarios de la estadística para detectar *outliers* era "tratar" estas observaciones anómalas que pueden distorsionar el análisis y modelado estadístico.

El término anomalía se utiliza en la ciencia de datos y áreas relacionadas con la detección de patrones y comportamiento inusual en un sentido más amplio que la definición de *outlier*. Como se mencionó anteriormente, una anomalía es una observación que se desvía notablemente de un patrón o comportamiento esperado en un conjunto de datos.

Así se podría afirmar que todo *outlier* es una anomalía, pero no necesariamente una anomalía es un *outlier*.

En el contexto de la ciencia de datos, notarás que cuando se emplean técnicas estadísticas para detectar anomalías, se empleará mas el término *outlier* que anomalía. Y cuando empleamos técnicas de aprendizaje de máquina para encontrar observaciones anormales, nos referiremos a estas como anomalías.

Cuando la anomalía es en una observación, se le denomina **anomalía puntual** (*Point anomaly* en inglés). Una anomalía puntual se define como una única observación (*data point*) que es inusual o anómala en comparación con el resto de las observaciones. Las anomalías puntuales suelen ser un valor extremo en una de las características (variable) de la observación. Estas anomalías son relativamente fáciles de identificar, ya que son puntos únicos que se encuentran lejos de donde se encuentran la mayoría de los datos. Este tipo de anomalía también se conoce como **anomalía global**.

Por ejemplo, un día del mes de abril puede presentar una temperatura muy alta frente a lo esperado. En este caso, la observación es el día, la variable es la temperatura y tenemos un día anómalo: una **anomalía puntual**. Consideremos otro ejemplo, en un conjunto de datos de ventas diarias, una anomalía puntual podría ser un día con una venta extremadamente alta o extremadamente baja en comparación con los otros días. Identificar estas anomalías puede ser importante para comprender las causas subyacentes de las fluctuaciones en las ventas y tomar medidas para abordarlas. En los Capítulos 2, 3 , 4 y 5 estudiaremos técnicas para detectar este tipo de anomalías. En el Capítulo 8 estudiaremos la técnica conocida como la distancia **kNN** que también es empleada para detectar **anomalías globales**.

Por otro lado, una **anomalía colectiva** (*collective anomaly* en inglés) es un conjunto de observaciones similares (en una o varias variables) que pueden considerarse

anómalas en conjunto cuando se comparan con el resto de los datos. Regresando a nuestro ejemplo de la temperatura, podríamos tener una semana entera en el mes de abril con temperaturas anormalmente altas. En ese caso tenemos una *anomalía colectiva*. Las observaciones con anomalía colectiva pueden ser también cada una de ellas anomalías puntuales, pero no siempre debe ser así. Por ejemplo, en el caso de las temperaturas diarias podemos tener una ola de calor, un solo día caliente para abril puede considerarse algo normal, pero varios días de temperaturas altas consecutivos pueden hacer que el evento se considere una anomalía. Análogamente, en el caso de las ventas diarias podríamos ver ventas durante una semana o un mes que sean anormalmente bajas. Esas ventas serían una anomalía colectiva.

Las anomalías colectivas, también son conocidas como **anomalías distribuidas** o **anomalías grupales**. Estas anomalías pueden ser difíciles de identificar, ya que no están presentes en puntos individuales, sino en patrones o grupos de datos.

Otro ejemplo de *anomalía colectiva*, que nos permite entender mejor el concepto, lo podríamos encontrar en datos del tráfico de una ciudad. Una anomalía colectiva podría ser un patrón de tráfico inusual que afecta a varias calles o manzanas de la ciudad al mismo tiempo. Identificar estas anomalías colectivas puede requerir técnicas como el análisis de clústering con el método **DBSCAN** que estudiaremos en el Capítulo 9 o el **Isolation Forest** que estudiaremos en el Capítulo 10.

Otra forma de clasificar las anomalías tiene que ver con el contexto. Las **anomalías contextuales**, también conocidas como **anomalías locales**, son observaciones que son anómalas en un contexto específico pero normales en otro contexto. Estas anomalías pueden ser difíciles de detectar, ya que requieren una comprensión profunda del dominio del problema y del contexto en el que se recopilaron los datos.

Por ejemplo, regresando al ejemplo de la temperatura, un día de 30 grados Celsius podría considerarse normal en un enero en una ciudad de una región tropical como la costa atlántica colombiana, pero anómala para una ciudad en el norte de los Estados Unidos en ese mismo mes. Identificar estas anomalías contextuales puede requerir no solo el análisis de los datos en sí, sino también información adicional sobre el entorno en el que se recopilaron los datos. En los Capítulos 8 estudiaremos la métrica **LOF** que permite la identificación de *anomalías locales*. En los Capítulos 9 y 10 también estudiaremos técnicas que permiten identificar este tipo de anomalías: **DBSCAN** y **Isolation Forest**.

Otra forma de clasificar[14] las anomalías es dependiendo del número de *features* que se consideren. Las **anomalías univariadas** son valores atípicos que se detectan considerando únicamente una variable (o característica). En el Capítulo 2 discutiremos más sobre este tipo de anomalías y estudiaremos métodos para detectar este tipo de ano-

[14]Otro tipo de anomalía específica para las muestras de series de tiempo son las **anomalías temporales**, también conocidas como **anomalías secuenciales**, son observaciones que se desvían significativamente de la tendencia temporal general de los datos. Por ejemplo, en una serie de ventas diarias, una anomalía temporal podría ser un pico de ventas inesperado en un día que normalmente tiene ventas bajas. Estas anomalías pueden ser difíciles de detectar, ya que requieren un análisis de series de tiempo para identificar patrones anómalos en el tiempo. En este libro no estudiaremos técnicas para detectar este tipo de anomalías.

malías. En los Capítulos 3 y 6 también estudiaremos técnicas para detectar *anomalías univariadas*.

Por otro lado, las **anomalías multivariadas** son valores atípicos que se detectan considerando múltiples variables simultáneamente. En este caso, la anomalía se identifica en el contexto de la relación entre dos o más dimensiones de datos. En la introducción del Capítulo 2 se presenta una discusión sobre la diferencia entre estas anomalías y las univariadas. En los Capítulos 4 y 5 discutimos técnicas que solo están diseñadas para detectar anomalías multivariadas. En los Capítulos 8 a 10 estudiaremos técnicas que permiten detectar tanto anomalías univariadas como multivariadas.

Anomalías y excepciones

En el contexto de la ciencia de datos, las anomalías y las excepciones son conceptos que se utilizan como sinónimos. Si bien ambos conceptos se emplean para describir eventos o casos que se desvían de lo que se considera normal o esperado, existen diferencias sutiles entre estos dos conceptos.

Una anomalía se refiere a una observación o evento que es "significativamente" diferente de la mayoría de los otros eventos en un conjunto de datos. Por otro lado, una excepción se refiere a un caso que se aparta de una regla, norma o criterio establecido. En este sentido, una excepción puede ser una anomalía, pero no necesariamente.

Las excepciones pueden ser eventos o situaciones que no siguen el patrón general o que no cumplen con ciertas condiciones predefinidas. Por ejemplo, en un sistema de detección de fraudes, una transacción que excede un cierto límite establecido podría considerarse una excepción, aunque no sea necesariamente una anomalía si es una transacción legítima.

Así, mientras que una anomalía se refiere a una observación inusual en un conjunto de datos, una excepción se refiere a un caso que se aparta de una regla o norma establecida, que puede o no ser una anomalía dependiendo del contexto y la interpretación. De acuerdo con la pregunta de negocio y los datos disponibles, en algunas situaciones la tarea de detección de anomalías se puede convertir en una tarea de detección de excepciones. En ambos casos, el objetivo es identificar eventos o situaciones que requieren una atención especial o una acción inmediata.

Las diferentes formas de clasificar las anomalías enumeradas anteriormente no son mutuamente excluyentes. Noten que unas anomalías contextuales podrían ser anomalías grupales y esas podrían ser univariadas o multivaridas. No existe una única taxonomía de las anomalías, estas definiciones de las anomalías son construidas típicamente para ilustrar el tipo de anomalía que un método puede detectar. Es más, las anomalías en campos específicos pueden tomar nombres especiales o clasificaciones especiales. Un ejemplo de esto es el campo de la detección de fraudes (Ver Sección 6.2).

En el mundo de los negocios, el análisis de anomalías puede aparecer en diferentes contextos. Por ejemplo, en los procesos productivos se quisiera detectar una anomalía en la planta de producción antes de que ocurra y solucionar los problemas antes de

que ocurran. Esto permite ahorrar mucho dinero. En el e-marketing se emplea la detección de anomalías para, por ejemplo, detectar cambios en el costo por clic (CPC) de una campaña, identificar un número inusualmente bajo de visitantes a una página web (podría haberse dañado un enlace a esa página). Se puede emplear para detectar puntos de venta con un comportamiento anormalmente bajo o alto frente a los de los demás puntos de venta. En el mantenimiento predictivo, es común detectar anomalías en equipos o maquinaria industrial para predecir fallas y realizar mantenimiento preventivo. Esto típicamente puede reducir costos y tiempos de inactividad. En el área de la salud se pueden detectar patrones anómalos en datos de pacientes para identificar enfermedades o condiciones médicas antes de que se manifiesten clínicamente.

1.3 Comentarios finales

En este capítulo, hemos explorado la importancia de la detección de anomalías en el ámbito de las organizaciones. Desde anomalías puntuales hasta anomalías colectivas, contextuales y multivariadas, estas desviaciones de lo "normal" pueden tener un impacto significativo en las organizaciones y sus operaciones.

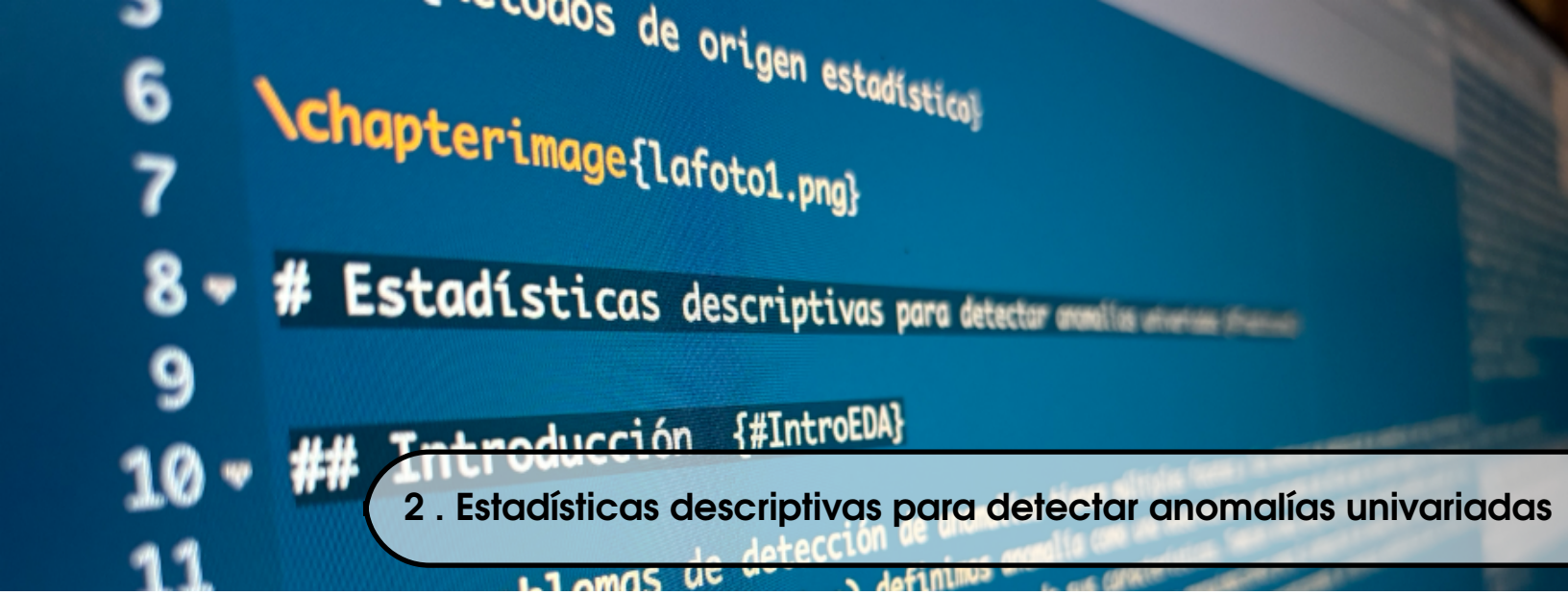
La detección de anomalías nos permite identificar eventos inusuales, patrones de comportamiento atípicos y tendencias anómalas en los datos. Esto nos brinda la oportunidad de comprender mejor los problemas, tomar decisiones informadas y, en muchos casos, tomar medidas preventivas.

En los próximos capítulos, exploraremos diferentes técnicas y herramientas para la detección de anomalías, así como su implementación en R. Aprenderemos cómo aplicar modelos de aprendizaje no supervisado para la detección de anomalías. Pero antes de entrar a estudiar los modelos de aprendizaje de máquina, estudiaremos la aproximación estadística a la detección de anomalías.

¡Sigue leyendo para descubrir más sobre este apasionante campo de la detección de anomalías en el mundo del *business analytics* y cómo puedes implementarlo en R!

Parte II

Métodos de origen estadístico



2 . Estadísticas descriptivas para detectar anomalías univariadas

2.1 Introducción

Los problemas de detección de anomalías tienen múltiples facetas y las técnicas de detección de anomalías son muy diferentes. En el Capítulo 1 definimos anomalía como una observación o un conjunto de ellas que no parece seguir el mismo patrón que el resto de los datos en una o varias de sus características. También vimos cómo los tipos de valores anómalos pueden ser muy diferentes. Adicionalmente hemos hablado de diferentes aproximaciones para la detección de anomalías que nacen desde la estadística y desde el aprendizaje de máquina. En este capítulo nos concentraremos en técnicas estadísticas para la detección de anomalías que se emplean en los análisis estadísticos de datos.

Antes de continuar, recordemos que en el contexto de la estadística un *outlier* es una observación que se desvía significativamente del resto de la muestra, ya sea en una o varias de sus características (*features* o variables)[1]. Como lo discutimos en el Capítulo 1, todo *outlier* es una anomalía, pero no toda anomalía es un *outlier*. Recuerda que el concepto de anomalía es más amplio que el de *outlier*, pues abarca no solo desviaciones puntuales extremas respecto de una distribución, sino también irregularidades dependientes del contexto y patrones colectivos. En este capítulo nos concentraremos en detectar anomalías puntuales univariadas con métodos estadísticos descriptivos. Así, en sentido estricto, en este capítulo estudiaremos métodos para detectar *outliers* en una sola variable que se emplean en el análisis estadístico de datos.

Cuando se realiza un análisis estadístico de datos, es común que se inicie con un análisis exploratorio de los datos. El análisis exploratorio de los datos, también conocido por su sigla en inglés **EDA** (*Exploratory Data Analysis*), implica examinar y analizar la muestra para comprender su estructura, características y patrones. El **EDA** se realiza utilizando una combinación de técnicas visuales y estadísticas, con el objetivo de:

[1] Estos valores pueden ser causados por errores en la medición, fraude, eventos excepcionales o simplemente por la naturaleza de los datos.

- Obtener una visión general de los datos: Esto incluye comprender el tipo de datos, la distribución de las variables, su tendencia central y volatilidad, así como la presencia de valores atípicos (*outliers*) y las relaciones entre las variables.
- Identificar patrones y tendencias: El **EDA** puede ayudar a descubrir patrones y tendencias interesantes en los datos que podrían no ser evidentes a simple vista.
- Formular hipótesis: El **EDA** puede ayudar a formular hipótesis sobre las relaciones entre las variables y los posibles factores que influyen en los resultados.
- Guiar el proceso de limpieza de los datos: El **EDA** puede ayudar a identificar y corregir problemas en los datos, como valores faltantes o inconsistencias, antes de realizar análisis estadísticos más formales.

De hecho, el primer objetivo está relacionado directamente con la tarea de detectar *outliers*[2]. Recuerda que en el análisis tradicional estadístico es importante detectar los valores atípicos, pues estos pueden generar que algunas técnicas estadísticas o estimadores no funcionen bien o no provean una buena idea de lo que está ocurriendo con la muestra. Por esto, el primer paso de cualquier análisis de anomalías típicamente inicia con un **EDA**. Al analizar los datos es importante distinguir entre los *outliers* univariados y multivariados.

Recuerda que en el Capítulo 1 definimos un *outlier univariado* como una observación que se desvía significativamente del resto en una sola variable. Estos datos pueden ser el resultado de errores de medición, errores de registro o eventos inusuales. Por otro lado, también definimos un *outlier multivariado* como una observación que se desvía significativamente del resto en múltiples variables simultáneamente. A diferencia de los *outliers* univariados que se destacan en una sola variable, los *outliers* multivariados rompen el patrón por completo. Esta observación anómala puede ser el resultado de una combinación de factores, como errores de medición, errores de registro o eventos inusuales que ocurren en varias variables al mismo tiempo. La detección de *outliers multivariados* es más compleja que la detección de *outliers* univariados, ya que requiere considerar las relaciones entre las variables. Por ejemplo, en la Figura 2.1 podemos identificar una observación atípica multivariada (en rojo) que se distancia del patrón de las otras observaciones, pero si consideramos cada una de esas variables de manera individual, no se observa un dato atípico. Ahí es cuando se presenta la dificultad para identificar esta anomalía.

[2]Los *outliers* se pueden considerar como sinónimo de anomalía puntual: una observación individual que, bajo un modelo o métrica de referencia, presenta un grado de rareza extremo.

Figura 2.1. Ejemplo de anomalías multivariadas

Fuente: elaboración propia.

Por ejemplo, si observamos para una de las 1000 tiendas de una organización unas ventas muy superiores a las demás en el año anterior, entonces estamos frente a un *outlier* univariado. Esto podría ser fruto de un error de registro, una estrategia de marketing excepcional o un fraude.

Por otro lado, consideramos un caso en el que tenemos dos *features* (variables) para cada cliente: unidades compradas y unidades devueltas (Ver Figura 2.1). De los miles de clientes de una organización, podríamos tener un cliente con compras inusualmente grandes y devoluciones de artículos relativamente grandes (Ver punto naranja en la Figura 2.1). Este cliente será un *outlier* multivariado[3]. También podemos encontrar un cliente que tiene compras muy parecidas al promedio y devoluciones ligeramente superiores al promedio (Ver punto rojo en la Figura 2.1). Este cliente se distancia del patrón que siguen los otros clientes y, por tanto, sería un *outlier* multivariado, si bien el cliente no presenta valores inusualmente diferentes al resto de clientes para cada variable. Lo que hace anómalo a este individuo es la combinación de valores para las dos variables. Esto podría indicar un fraude, un error en el registro o un problema con el producto o servicio recibido por ese cliente.

En este capítulo nos concentraremos en detectar anomalías univariadas; en el Capítulo 4 estudiaremos las técnicas estadísticas basadas en métricas para encontrar *outliers* multivariados.

[3]Si consideramos por separado cada variable para esa observación, la observación también será un *outlier* univariado (con respecto a las dos variables).

Regresando al **EDA**, típicamente un **EDA** implica por lo menos los siguientes pasos:

- Graficar los datos: Hacer gráficas de los datos es una herramienta importante para el inicio de un **EDA** pues nos permite entender gráficamente lo que ocurre con nuestra muestra. Típicamente se emplean gráficos como histogramas para ver la distribución de cada variable y, en especial, los *boxplot*[4] para hacer visibles los *outliers*. Así mismo, los diagramas de dispersión se suelen emplear para detectar los valores atípicos multivariados.
- Calcular estadísticas descriptivas para cada variable (análisis univariado): En este paso, se analiza cada variable y dependiendo del tipo (numérica, categórica, etc.) se calculan estadísticas de tendencia central, dispersión, posición (por ejemplo como los cuartiles) y forma.

En este capítulo estudiaremos algunas de las métricas de origen estadístico más populares empleadas en un **EDA** para detectar valores atípicos y repasaremos las dos gráficas más comunes cuando se está indagando por valores atípicos: el histograma y el *boxplot*.

2.2 Aproximación gráfica

Los gráficos más empleados para detectar *outliers* univariados son los histogramas y los *boxplots*. Recuerden que un histograma agrupa los datos en intervalos pequeños que se miden típicamente en el eje horizontal, y en el eje vertical se representa la frecuencia de aparición de las observaciones dentro de los límites del intervalo.

No obstante, el histograma presenta la distribución de los datos, el *boxplot* fue diseñado por Tukey (1970) para detectar *outliers*[5]. El *boxplot* o diagrama de cajas[6] permite observar el primer, segundo (mediana) y tercer cuartil, la distancia intercuartílica (distancia entre el primer y tercer cuartil de los datos) y la existencia o no de datos atípicos.

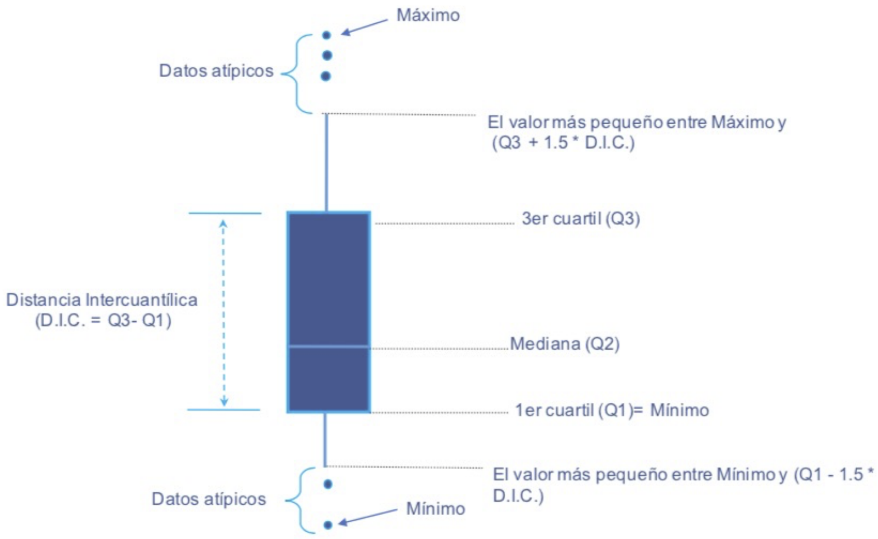
La caja central del *boxplot* se extiende desde el primer cuartil (Q1) hasta el tercer cuartil (Q3) (Ver Figura 2.2). De esta manera, el 50% de los datos están contenidos en la caja. Dentro de la caja, se traza una línea que marca la mediana (Q2) de los datos (Ver Figura 2.2). La longitud de la caja se conoce como **Distancia intercuartílica** (**DIC**) o **rango intercuartílico** (**IQR**). Por otro lado, tenemos los bigotes (*Whiskers* en inglés) que son las líneas que se extienden de los extremos de la caja. El bigote superior se extiende desde el extremo superior de la caja (Q3) a lo que sea menor entre el valor máximo de las observaciones o 1.5 veces el **IQR** (o **DIC**). Y el otro bigote sale de la parte inferior de la caja (Q2) hasta lo que sea más pequeño entre el valor mínimo de los datos y 1.5 veces el **IQR** (o **DIC**). Los bigotes indican la variabilidad de los datos fuera de los cuartiles superior e inferior. Si existen observaciones que se encuentren por fuera de los bigotes, estas observaciones se representan con un punto y se consideran valores atípicos (Ver Figura 2.2).

[4] También conocidos como diagrama de cajas y bigotes.

[5] Siete años después, Tukey publica el libro que se divulga ampliamente y populariza esta gráfica (Ver Tukey (1977))

[6] Este gráfico también es conocido como el diagrama de cajas y bigotes.

Figura 2.2. Elementos de un Boxplot



Fuente: tomado de Alonso y Largo (2024).

En distribuciones asimétricas, el *boxplot* puede "marcar" muchos puntos no anómalos como valores atípicos. Hubert y Vandervieren (2008) propuso un ajuste al *boxplot* para resolver este problema: el *boxplot* ajustado por asimetría. Este *boxplot* corrige la asimetría utilizando una medida robusta en presencia de asimetría para determinar el "umbral" para los valores atípicos. Esto implica cambiar el límite de 1.5 la distancia intercuartílica por una medida corregida. Para un mayor detalle, ver Hubert y Vandervieren (2008).

2.3 Métricas para detectar outliers en una variable

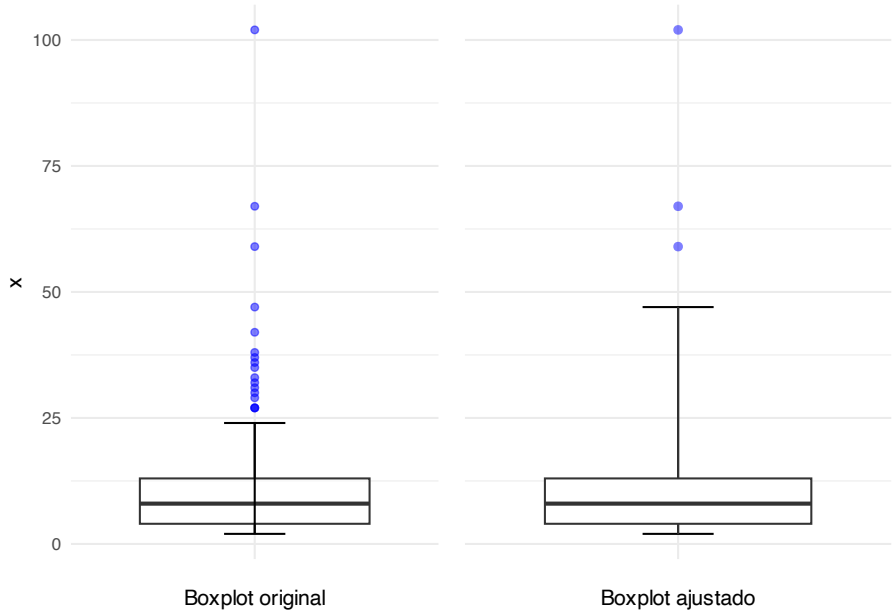
Una aproximación complementaria a la aproximación gráfica para detectar anomalías univariadas es emplear estadísticas o métricas que permitan estandarizar los datos o construir "umbrales" o intervalos que sirvan como "fronteras" para determinar a partir de qué valor se determinará que una observación es o no anomalía. A continuación discutiremos 4 aproximaciones diferentes.

2.3.1 Uso de percentiles

Un método utilizado para la detección de valores atípicos se basa en los percentiles. Para este método, todas las observaciones que se encuentren fuera del intervalo formado por los percentiles $2{,}5$ y $97{,}5$ se considerarán posibles valores atípicos. Dependiendo del caso, también se pueden considerar otros percentiles como límites para construir el intervalo; por ejemplo, los percentiles 1 y 99, o 5 y 95.

Un gran problema de este método es que siempre encontraremos *outliers* con esta

Figura 2.3. Comparación del boxplot original y el ajustado según Hubert y Vandervieren (2008)



Fuente: elaboración propia.

aproximación, pues siempre podremos encontrar las observaciones menores al percentil 1 y al percentil 99. Por ejemplo, si se cuenta con 100 observaciones, una observación siempre estará por debajo del primer percentil y otra estará por encima del percentil 99. Esto no necesariamente implica que estas observaciones sean atípicas. Esta aproximación tendrá que ser empleada con mucho cuidado.

2.3.2 Z-score

El *z-score*, también conocido como puntaje Z o índice de desviación estándar, es una medida estadística que nos permite comparar una observación con la media y la desviación estándar de la muestra. Es decir, el *z-score* estandariza los datos restándole la media ($\bar{X}$) y dividiéndolos por la desviación estándar (S). De esta manera, las puntuaciones z son el número de desviaciones estándar por encima y por debajo de la media de cada valor.

Formalmente el puntaje z para la observación i de la variable X es:

$$Z_i = \frac{X_i - \bar{X}}{S} \tag{2.1}$$

Cuanto más se aleja de cero la puntuación z de una observación, más inusual es. Un valor de corte "estándar" para encontrar valores atípicos es las puntuaciones Z mayores que 3 y menores que -3. Este umbral tiene origen en la distribución normal, en

la que se espera que más allá de tres desviaciones estándar solo se encuentren 0.27 % de las observaciones. Es decir, si los datos vienen de una distribución normal, es poco probable obtener una observación con un z-score mayor en valor absoluto que 3.

Sin embargo, si los datos no siguen la distribución normal, esta aproximación podría no ser conveniente. Por otro lado, la misma presencia del valor atípico distorsiona las puntuaciones z, ya que infla la media y la desviación estándar. En otras palabras, si una muestra contiene valores atípicos, los valores z están sesgados de tal forma que parecen ser menos extremos (más cercanos a cero).

2.3.3 Rango intecuartílico

La prueba de Tukey, también conocida como el método de los "bigotes" o el criterio del rango intercuartílico, es una técnica estadística utilizada para identificar valores atípicos en una variable o característica (datos univariados). Funciona creando un intervalo empleando la distancia entre el primer y tercer cuartil de los datos (rango intercuartílico) (**IQR**). El intervalo se construye restándole al primer cuartil ($q_{0,25}$) 1.5 veces la **IRQ** y sumándole al tercer cuartil ($q_{0,75}$) 1.5 veces la **IQR**. Si un valor está por fuera del intervalo, se considera un outlier.

En otras palabras, el intervalo I_T para esta prueba se construye de la siguiente manera:

$$I_T = [q_{0,25} - 1{,}5IQR; q_{0,75} + 1{,}5IQR] \tag{2.2}$$

Así, cualquier valor por encima del límite superior es considerado un outlier; de manera similar, las observaciones por debajo del límite inferior son consideradas *outliers*. La prueba de Tukey es útil para identificar valores atípicos en un conjunto de datos de manera rápida y corresponde al mismo análisis que se realizaría empleando un *boxplot*.

2.3.4 Método de Hampel

El método de Hampel[7] (en inglés se conoce como *Hampel Outlier Threshold*) es un método estadístico no paramétrico para la detección de *outliers* que se basa en la idea de comparar cada observación con sus vecinos más cercanos. Los puntos de datos que se desvían sustancialmente de sus vecinos se identifican como **outliers**. Para identificar si la desviación a los vecinos es grande se crea el intervalo I_H:

$$I_H = [median - k \cdot MAD; median + k \cdot MAD] \tag{2.3}$$

donde MAD corresponde a la desviación absoluta de la mediana (*Median Absolute Deviation* en inglés) que se define como la mediana de las desviaciones absolutas con respecto a la mediana de los datos ($\tilde{X}$). Es decir,

$$MAD = median(|X_i - \tilde{X}|) \tag{2.4}$$

[7] Este método fue propuesto por Hampel (1974).

Típicamente, se emplea $k = 3$ en (2.3) siguiendo la denominada regla de Pearson[8]. En últimas k es un factor que determina la sensibilidad del método a los outliers. Si k es demasiado pequeño, esto puede conducir a la detección de muchos *outliers*; mientras que si k es muy grande puede que no se encuentren *outliers*.

2.3.5 Metricas que emplean estimadores robusto de varianza

Otra forma de encontrar *outliers* es crear umbrales o puntos de corte que empleen estimadores de varianza y media robustos a la presencia de valores atípicos. En este caso, el primer paso es encontrar una medida robusta de la raíz cuadrada media ($RRMS$ por el término *robust root mean square*). En general la raíz cuadrada media (RMS) para una variable X_i se define como:

$$RMS = \sqrt{\bar{X}^2 + S^2} \tag{2.5}$$

donde $\bar{X}$ representa la media y S^2 es la varianza. La raíz cuadrada media (RMS), también conocida como valor cuadrático medio o valor eficaz, es una estadística descriptiva de volatilidad que permite calcular la variación promedio.

Por otro lado, recordemos que la existencia de valores atípicos implica que la media y la varianza se verán afectadas por esos valores atípicos. La $RRMS$ emplea estimadores robustos de la tendencia central y de la varianza para mitigar el impacto de los valores atípicos en la media y varianza. Así, el $RRMS$ será:

$$RRMS = \sqrt{med(X)^2 + est.var(X)} \tag{2.6}$$

donde $med(X)$ es la mediana y $est.var(X)$ es un estimador robusto para la varianza. Por ejemplo, el $est.var(X)$ puede ser la desviación absoluta de la mediana (MAD) como se define en (2.4).

Empleando el $RRMS$ se pueden escalar los datos[9] y así poderlos comparar con una distribución estadística. Así, podríamos discriminar entre los valores atípicos.

Esta aproximación implica construir un intervalo para los valores que no se consideran atípicos (como lo realizamos en las aproximaciones anteriores). En este caso el umbral superior estará dado por:

$$Umbral_{Superior} = med(X/RRMS) + F^{-1}(p) \cdot RRMS \tag{2.7}$$

donde $F()$ es una función de distribución acumulativa como por ejemplo la distribución t o normal. Así, $F^{-1}()$ es la función cuantil. Y p es la probabilidad de $F^{-1}()$ a partir de la cual deseamos definir el corte para los valores críticos. De manera similar se puede encontrar el umbral inferior. Así, una vez que se tienen los umbrales, podemos constatar que las observaciones superan dichos umbrales y, por tanto, se podrían catalogar como datos atípicos.

[8] Esta regla también es conocida como la regla de Tukey-Pearson. La regla de Pearson se basa en la idea de comparar la distancia entre el punto de datos que se está evaluando y la mediana con la distribución de las distancias entre los puntos de datos dentro de la ventana.

[9] Algo similar a lo que se desea hacer con el z-score, pero en este caso teniendo en cuenta que para la escalada necesitamos medidas robustas a la presencia de *outliers*.

2.4 Empleando R para encontrar anomalías univariadas

En un **EDA**, inicialmente, los valores atípicos se pueden detectar evaluando los valores que son lógicamente imposibles. Por ejemplo, un ingreso para un cliente negativo. Para lograr identificar un *outlier* debemos comparar la observación con otras observaciones realizadas sobre el mismo fenómeno, pues es posible que la existencia misma del valor atípico pueda deberse a la variabilidad inherente al fenómeno que se observa. Adicionalmente, estas observaciones atípicas pueden aparecer debido a un error de medición o de codificación. Por tanto, es importante distinguir entre un *outlier* y un valor errado.

En esta sección implementaremos las métricas discutidas anteriormente en R. Para esto, emplearemos una base de datos de un banco comercial alemán que contiene información de clientes a los que se les ha otorgado un crédito y se han calificado como buenos o malos deudores. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalias). La base de datos contiene las siguientes variables:

- `edad` (numérica) = edad,
- `residente` (numérica) = años que lleva residiendo en Alemania,
- `propiedad.hogar` = con las siguientes categorías: "rent", "own","for free",
- `trabajo` = con las siguientes categorías: "unemployed/ unskilled - non-resident", "unskilled - resident","skilled employee / official", "management/ self-employed/", "highly qualified employee/ officer",
- `dependientes` (numérica)= número de personas que dependen del cliente
- `trabajador.extranjero` = trabajador extranjero con las siguientes categorías: "yes", "no",
- `empleado` = tiempo que lleva empleado con las siguientes categorías: "unemployed", "less than 1 year", "equal or more than 1 and less than 4 years"," equal or more than 4 and less than 7 years", "equeal or more than 7 years"
- `estadocivil.genero` = con las siguientes categorías: "male: divorced/separated", "female: divorced/separated/married", "male: single", "male: married/widowed", "female: single".

Por otro lado, se cuenta con las siguientes variables relacionadas con el banco:

- `numero.creditos` (numérica) = número de créditos existentes con el banco,
- `porcentage.disponible` (numérica)= porcentaje del ingreso disponible para pagar cuotas de créditos,
- `cuenta.corriente` (numérica)= estado de la cuenta corriente actual con categoría de respuesta: "less than 0 DM", "0 or less than 200 DM", "equal or greater than 200 DM / salary assignments for at least 1 year", "no checking account",
- `proposito.credito`: propósito del crédito con las siguientes categorías de respuesta: "car (new)", "car (used)", "furniture/equipment","radio/television", "domestic appliances", "repairs", "education", "vacation", "retraining", "business", "others",
- `monto.credito` (numérica) = monto del crédito,
- `cuenta.ahorros` = estado de la cuenta de ahorros con las siguientes categorías: "less than 100 DM", "equal or greater than 100 and less than 500 DM", "equal or

greater than 500 and less than 1000 DM", "greater than 1000 DM", "unknown/no savings account".

Finalmente, se cuenta con la variable:

- `historial.crediticio` = con categorías de respuestas: "no credits taken/ all credits paid back duly", "all credits at this bank paid back duly", "existing credits paid back duly till now", "delay in paying off in the past", "critical account/ other credits existing (not at this bank)".

Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Miremos rápidamente los datos cargados empleando la función **glimpse()** del paquete *dplyr* (Wickham et al., 2021).

```
library(dplyr)
glimpse(german)
```

```
## Rows: 1,000
## Columns: 14
## $ cuenta.corriente      <fct> less than 0 DM, 0 or less than 200 DM, no checki~
## $ historial.crediticio  <fct> critical account/ other credits existing (not at~
## $ proposito             <fct> radio/television, radio/television, education, f~
## $ monto.credito         <int> 1169, 5951, 2096, 7882, 4870, 9055, 2835, 6948, ~
## $ cuenta.ahorros        <fct> unknown/ no savings account, less than 100 DM, l~
## $ porcentaje.disponible <int> 4, 2, 2, 2, 3, 2, 3, 2, 2, 4, 3, 3, 1, 4, 2, 4, ~
## $ estadocivil.sexo      <fct> male : single, female : divorced/separated/marri~
## $ residente             <int> 4, 2, 3, 4, 4, 4, 4, 2, 4, 2, 1, 4, 1, 4, 4, 2, ~
## $ edad                  <int> 67, 22, 49, 45, 53, 35, 53, 35, 61, 28, 25, 24, ~
## $ propiedad.hogar       <fct> own, own, own, for free, for free, for free, own~
## $ numero.creditos       <int> 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, ~
## $ ocupacion             <fct> skilled employee / official, skilled employee / ~
## $ dependientes          <int> 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ trabajador.extranjero <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes~
```

Los datos han sido cargados satisfactoriamente, y la clase de cada variable concuerda con su descripción. Ahora procedamos a realizar el **EDA**.

2.4.1 Graficandolos datos

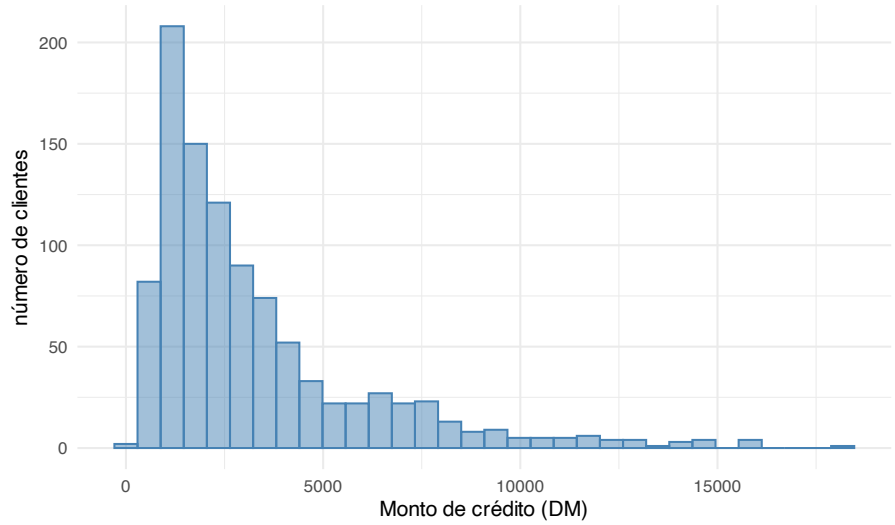
Empecemos nuestro análisis graficando los datos y para hacer el ejercicio más ágil, concentremos nuestra atención en la variable `monto.credito`[10].

[10]En un **EDA** es necesario replicar el análisis que se presenta en esta sección a todas las demás variables. El análisis se debe realizar variable por variable. Por razones de espacio no realizamos dicho análisis en esta obra.

Inicialmente, vamos a recurrir a los histogramas para detectar valores atípicos. Empleemos el paquete *ggplot2* (Wickham, 2016) para construir el histograma[11].

```
source = "Cálculos propios."
library(ggplot2)

german %>%
    ggplot(aes(x = monto.credito)) + geom_histogram(bins =
    ↪  round(sqrt(length(german$monto.credito))),
    fill = "steelblue", color = "steelblue", alpha = 0.5) + xlab("Monto de
    ↪  crédito (DM)") +
    ylab("número de clientes") + theme_minimal()
```

Figura 2.4. Histograma de la variable Monto de crédito



Fuente: elaboración propia.

En el histograma vemos que existen unas observaciones de montos de créditos por encima de los 15 mil DM, relativamente grandes con respecto a las demás observaciones de los demás clientes (Ver las barras en el lado derecho de la Figura 2.4). Esto ya nos da unas sospechas de la existencia de valores atípicos (univariados) en esta variable. También podemos ver que los datos presentan asimetría positiva o sesgo a la derecha[12].

Ahora construyamos el respectivo *boxplot* empleando el paquete *ggplot2*

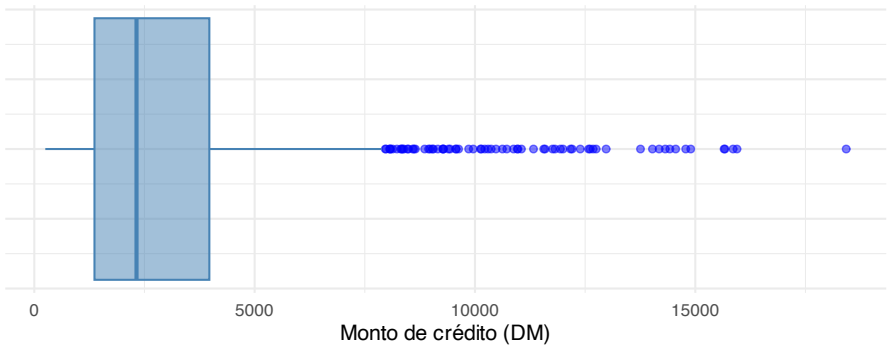
[11] Para el detalle de cómo funciona el paquete y este gráfico puedes ver la sección 3.1 de Alonso y Largo (2023). Puedes acceder a esa sección con el siguiente enlace: https://www.icesi.edu.co/editorial/empezando-visualizar-2ed-web/dist.html#histograma.

[12] Para una discusión sobre el tema puedes consultar la sección 15.6 de Alonso (2024) empleando el siguiente enlace: https://www.icesi.edu.co/editorial/modelo-clasico-web/Estadistica.html#varianza-y-momentos-alrededor-de-la-media-de-una-variable-aleatoria.

(Wickham, 2016)[13]. El código que genera la Figura 2.5 es el siguiente:

```
german %>%
    ggplot(aes(x = monto.credito)) + geom_boxplot(fill = "steelblue",
    ↪  outlier.colour = "blue",
    alpha = 0.5, color = "steelblue") + xlab("Monto de crédito (DM)") +
    ↪  ylab("") +
    theme_minimal() + theme(axis.text.y = element_blank(), axis.ticks =
    ↪  element_blank(),
    axis.title.y = element_blank())
```

Figura 2.5. Boxplot de la variable Monto de crédito



Fuente: elaboración propia.

Como se discutió anteriormente, el gráfico de caja y bigotes permite visualizar una variable cuantitativa y permite mostrar estadísticas resumen tales como: mínimo, mediana, primer ($q_{0,25}$) y tercer ($q_{0,75}$) cuartil, máximo y cualquier observación que se clasifique como un valor atípico sospechoso utilizando el criterio de rango intercuartil (IQR).

En este caso, los datos atípicos parecen muchos (Ver puntos azules en la Figura 2.5); son 72 observaciones[14]. Recordemos que este resultado puede estar influenciado por la asimetría de la distribución.

Siguiendo a Hubert y Vandervieren (2008) podemos ajustar al *boxplot* para resolver este problema. Esto lo podemos realizar empleando la función **adjbox()** del paquete *robustbase* (Maechler et al., 2024). Esta función solo necesita como argumento la variable a la que se le generará el *boxplot* ajustado. Así el *boxplot* ajustado se puede crear empleando el siguiente código:

```
# Crear boxplot ajustado según Hubert y Vandervieren (2008)
adjbox(german$monto.credito)
```

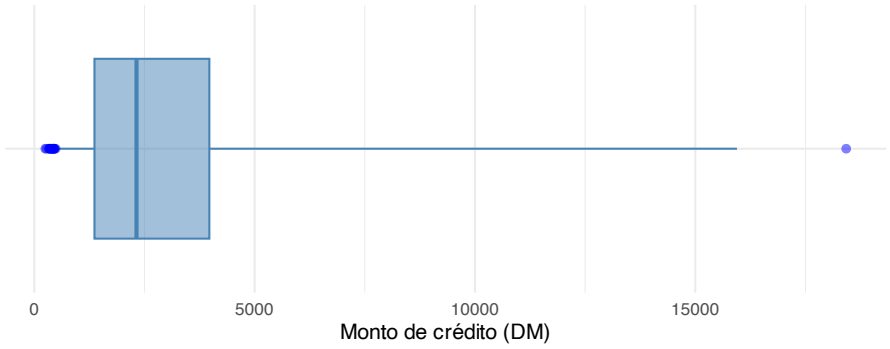
[13]Para el detalle de cómo funciona el paquete y esta gráfica puedes ver la sección 3.3 de Alonso y Largo (2023). Puedes acceder a esa sección con el siguiente enlace: https://www.icesi.edu.co/editorial/empezando-visualizar-2ed-web/dist.html#box.

[14]En la Sección 2.4.2.3 veremos en detalle cómo encontrar las observaciones que corresponden a anomalías.

Esta función genera un *boxplot* ajustado empleando la base de R. Con un poco más de trabajo y empleando la función **adjbox_stats()** del mismo paquete *robustbase* podemos construir un *boxplot* ajustado empleando el paquete *ggplot2*. El siguiente código genera la Figura 2.6.

```
library(robustbase)
library(ggplot2)

adjbox_stats <- adjboxStats(german$monto.credito)$stats


ggplot(data.frame(german$monto.credito), aes(x = german.monto.credito, y = ""))
↪  +
    geom_boxplot(xmin = adjbox_stats[1], xmax = adjbox_stats[5], middle =
    ↪  adjbox_stats[3],
        upper = adjbox_stats[4], lower = adjbox_stats[2], outlier.shape = NA,
        ↪  fill = "steelblue",
        outlier.colour = "blue", alpha = 0.5, color = "steelblue") +
        ↪  geom_point(data = subset(data.frame(german$monto.credito),
    german.monto.credito < adjbox_stats[1] | german.monto.credito >
↪   adjbox_stats[5]),
    col = "blue", size = 2, shape = 16, alpha = 0.5) + xlab("Monto de crédito
    ↪  (DM)") +
    ylab("") + theme_minimal() + theme(axis.text.y = element_blank(),
    ↪  axis.ticks = element_blank(),
    axis.title.y = element_blank())
```

Figura 2.6. Boxplot ajustado según Hubert y Vandervieren (2008) para la variable Monto de crédito



Fuente: elaboración propia.

El *boxplot* ajustado muestra 19 datos atípicos tanto en la cola inferior como en la cola superior (Ver puntos azules en la Figura 2.6). De hecho, solo se encuentra un solo *outlier* grande y el resto son valores bajos[15].

[15]En la Sección 2.4.2.3 veremos en detalle cómo encontrar las observaciones que corresponden a ano-

2.4.2 Métricas para detectar outliers

Recurrir a la estadística descriptivas y a las gráficas de los datos permite descubrir múltiples aspectos en estos tales como: patrones y en algunas ocasiones *outliers*. Sin embargo, es necesario recurrir a métricas o estadísticas que permitan estandarizar los datos o construir "umbrales" o intervalos que sirvan como "fronteras" para determinar a partir de qué valor se determinará que una observación es o no atípica. A continuación estudiaremos cómo implementar en R las cuatro métricas discutidas anteriormente.

Para calcular las estadísticas descriptivas hay varias opciones. Por ejemplo, podemos calcular las estadísticas descriptivas, variable por variable, con la función **summary()** de la base de R.

Por ejemplo,

```
summary(german$monto.credito)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     250    1366    2320    3271    3972   18424
```

Otra opción para calcular las estadísticas descriptivas de una variable es emplear la función **describe()** del paquete *psych* (William Revelle, 2023). Por ejemplo:

```
# install.packages('psych')
library(psych)
psych::describe(german$monto.credito)
```

```
##    vars    n    mean      sd median trimmed     mad min   max range skew
## X1    1 1000 3271.26 2822.74 2319.5 2754.57 1627.15 250 18424 18174 1.94
##    kurtosis    se
## X1     4.25 89.26
```

Nota que después de la mediana se presenta una estadística descriptiva que se denomina "trimmed". Esta corresponde a la media truncada; es decir, es la media si se descartan un porcentaje de los datos más altos y de los más bajos. En el caso de la función **describe()**, del paquete *psych*, por defecto calcula la media truncada descartando el 10% superior e inferior de los datos (el 5% más bajo y el 5% más alto). Comparando la media y la media truncada, puede evidenciarse una diferencia relativamente grande entre ellas, mostrando la posible existencia de valores anormalmente grandes. Si se quisiera recortar una proporción diferente, se puede hacer empleando el argumento **trim** de esa función. Por ejemplo, si se desea recortar solo el 5% de los datos extremos, el código sería:

```
psych::describe(german$monto.credito, trim = 0.05)
```

Puedes constatar que la media, truncando el 30% de las observaciones (15% más altas y 15% más bajas), es de 2387.23. Y si se recorta el 20%, es 2495.12. Estos dos números son diferentes pero no tanto como ocurre en el caso en que se elimina solo el 10%

malías empleando este *boxplot* ajustado.

de los datos, 2754.57. Claramente hay unos datos anormalmente altos que jalonan la media.

2.4.2.1 Usando percentiles

Encontrar las observaciones que superan (o están por debajo de) un percentil se puede hacer fácilmente empleando el paquete *dplyr* y la función **quantile()** de la base de R. Supongamos que queremos detectar cuáles son los montos de crédito más altos. En especial queremos encontrar los clientes tal que el 99.5% de los clientes tengan montos de crédito inferiores. Esto se puede hacer de la siguiente manera[16]:

```
german %>%
    select(monto.credito) %>%
    filter(monto.credito > quantile(monto.credito, 0.995))
```

```
##   monto.credito
## 1         15945
## 2         15653
## 3         15857
## 4         15672
## 5         18424
```

Nota que por construcción, dado que tenemos 1000 observaciones, las observaciones que están por encima del percentil 99.5 serán 5.

2.4.2.2 z-score

El *z-score* para cada observación para un *feature* no es más que el resultado de estandarizar la variable. Esto se puede realizar empleando la función **scale()** de la base de R. El siguiente código produce los z-scores para la variable `monto.credito` para cada observación:

```
z_scores <- as.data.frame(scale(german$monto.credito))
names(z_scores) <- "score"
```

Y podemos encontrar las observaciones que tengan un *z-score* menor a -3 de la siguiente manera:

```
z_scores %>%
    filter(score < -3)
```

```
## [1] score
## <0 rows> (or 0-length row.names)
```

Como se puede observar, no se encuentran valores atípicamente bajos con esta aproximación. De hecho, esto no es una sorpresa, pues el *z-score* tiene en mente una variable que provenga de una distribución normal (simétrica), lo cual no es lo adecuado para nuestro caso.

[16]Para entender cómo funciona el paquete *dplyr* y cómo se filtran observaciones, puedes consultar la Sección 2.1 de Alonso (2022). Ese documento lo puedes consultar en el siguiente enlace: https://www.icesi.edu.co/editorial/empezando-transformar-web/filtrar.html#filtrar-1.

De manera similar, encontramos las observaciones con puntaje z por encima de 3.

```
out_obs <- z_scores %>%
    filter(score > 3)

dim(out_obs)
```

```
## [1] 25  1
```

En este caso encontramos 25 observaciones cuyo puntaje z supera el umbral de 3. Nuevamente, esto parece algo excesivo.

2.4.2.3 Rango intercuartílico

Recordemos que el criterio IQR para detectar datos anómalos implica marcar todas las observaciones por encima de $q_{0,75} + 1{,}5IQR$ o por debajo de $q_{0,25} - 1{,}5IQR$ como valores atípicos (Ver puntos azules en la Figura 2.5).

Es posible extraer los valores de los posibles valores atípicos que se graficaron en el *boxplot* (Ver Figura 2.5 empleando la función **boxplot.stats()** del paquete *grDevices* (R Core Team, 2023). La función típicamente incluye los siguientes dos argumentos:

boxplot.stats(x, coef = 1.5)

donde:

- **x**: Un vector de clase **numeric** que contiene las observaciones graficadas en el boxplot.
- **coef**: corresponde a la extensión de los "bigotes" del *boxplot* en términos del IQR. Por defecto, **coef = 1.5**. Es decir, se emplea 1.5 veces la distancia intercuartílica.

Esta función provee varios resultados importantes para construir la gráfica del *boxplot* que son guardados en diferentes compartimientos (*slots*); entre los cuales se destaca el *slot* `out` en el que se guardan los valores que se encuentran por encima de $q_{0,75}+1{,}5IQR$ o por debajo de $q_{0,25} - 1{,}5IQR$. En nuestro caso, los valores de los *outliers* de la variable `monto.credito` pueden ser encontrados con el siguiente código:

```
# install.packages('grDevices')
library(grDevices)

boxplot.stats(german$monto.credito)$out
```

```
##  [1]  9055  8072 12579  9566 14421  8133  9436 12612 15945 11938  8487 10144
## [13]  8613  9572 10623 10961 14555  8978 12169 11998 10722  9398  9960 10127
## [25] 11590 13756 14782 14318 12976 11760  8648  8471 11328 11054  8318  9034
## [37]  8588  7966  8858 12389 12204  9157 15653  7980  8086 10222 10366  9857
## [49] 14027 11560 14179 12680  8065  9271  9283  9629 15857  8335 11816 10875
## [61]  9277 15672  8947 10477 18424 14896 12749 10297  8358 10974  8386  8229
```

Recurriendo al uso de la función *which*() es posible obtener el número de la fila correspondiente a los valores atípicos; es decir el número de la observación. En este caso el código es:

```
out <- boxplot.stats(german$monto.credito)$out
out_ind <- which(german$monto.credito %in% c(out))
out_ind
```

```
##  [1]   6  18  19  58  64  71  79  88  96 106 131 135 137 181 206 227 237 269 273
## [20] 275 286 292 296 305 334 374 375 379 382 396 403 418 432 451 492 497 510 526
## [39] 550 564 616 617 638 646 654 658 673 685 715 737 745 764 772 806 809 813 819
## [58] 829 833 855 882 888 896 903 916 918 922 928 946 954 981 984
```

Con esta información, es posible volver fácilmente a las filas específicas del conjunto de datos para verificarlas o imprimir todas las variables para estos valores atípicos. Según el método, se tiene un total de 72 observaciones atípicas.

```
out_dataset <- german[out_ind, ]
```

Por otro lado, recordemos que esta aproximación puede generar muchos valores atípicos en presencia de una muestra con sesgo. En la Figura 2.6 aplicamos la técnica propuesta por Hubert y Vandervieren (2008) para ajustar el *boxplot* a muestras con sesgo. En ese caso encontramos 19 datos atípicos tanto en la cola inferior como en la cola superior (Ver puntos azules en la Figura 2.6).

Para conocer los valores de las observaciones que se considerarían *outliers* con este ajuste (rango intercuartílico ajustado) lo podemos hacer con la función **adjboxStats()** del paquete *robustbase* (Wickham et al., 2021). Esta función es muy similar a la función **boxplot.stats()** que acabamos de emplear. En este caso solo necesitamos un argumento: los datos que se grafican en el *boxplot*. El siguiente código permite encontrar los valores de las observaciones atípicas empleando la corrección de Hubert y Vandervieren (2008):

```
adjboxStats(german$monto.credito)$out
```

```
##  [1]   426   409   458   392   339   338   433   276   362   343   448   368
## [13]   385   433   250   428   484 18424   454
```

De manera similar a lo que hicimos con la IRQ sin ajuste, es posible obtener el número de la fila correspondiente a los valores atípicos con el siguiente código:

```
out <- adjboxStats(german$monto.credito)$out
out_ind <- which(german$monto.credito %in% c(out))
out_ind
```

```
##  [1]  27  28  40 112 158 178 250 310 380 459 472 494 591 722 726 751 812 916 965
```

Otra forma de aplicar esta aproximación es empleando la función *tukey_outlier()** del paquete *funModeling* (Casas, 2024). Tú puedes mirar en la ayuda cómo emplear esta función.

2.4.3 Método de Hampel

El método de Hampel se puede implementar en R con el paquete *funModeling* (Casas, 2024) y la función **hampel_outlier()**. Esta función solo necesita dos argumentos. El argumento **input** que corresponde al vector de datos cuantitativos y el argumento **k_mad_value** que corresponde al multiplicador de la desviación absoluta mediana; es decir, k en la expresión (2.3). El valor por defecto de este segundo argumento es **k_mad_value = 3** ($k = 3$).

El siguiente código genera los umbrales de Hampel (Ver expresión (2.3)) para la variable `monto.credito`:

```
# install.packages('funModeling')

# Cargar el paquete
library(funModeling)

h_outliers <- hampel_outlier(input = german$monto.credito)
h_outliers
```

```
## bottom_threshold    top_threshold
##         -2561.96          7200.96
```

El siguiente código permite identificar el número de las observaciones que no se encuentran en el intervalo de Hampel; es decir, las que superan el umbral para la variable `monto.credito` y por tanto se consideran *outliers*:

```
out_ind <- which(german$monto.credito < h_outliers[1] | german$monto.credito >
↪  h_outliers[2])
out_ind
```

```
##   [1]   4   6  18  19  49  58  64  71  79  88  96 106 109 114 131 135 137 154
##  [19] 164 176 181 206 227 228 237 256 269 273 275 286 288 292 295 296 305 333
##  [37] 334 374 375 376 379 382 388 396 403 412 418 432 451 468 492 497 510 526
##  [55] 539 550 564 616 617 638 646 651 654 658 673 685 715 716 737 745 764 772
##  [73] 797 805 806 809 813 816 819 829 833 855 869 871 881 882 888 890 896 903
##  [91] 916 918 922 928 946 954 972 974 981 984
```

En este caso se encuentran 100 observaciones marcadas como *outliers* por el método de Hampel. Y con el siguiente código podemos ver los valores de dichas observaciones para la variable `monto.credito`:

```
out_obs <- german$monto.credito[out_ind]
out_obs
```

2.4.4 Métricas que emplean estimadores robusto de varianza

El paquete *hotspots* (Darrouzet-Nardi, 2018) permite identificar valores atípicos empleando estimadores robustos a la presencia de datos atípicos para la media (mediana) y la varianza (MAD) y comparándolos con lo esperado con una distribución

(como la distribución t y normal).

La función **outliers()** del paquete *hotspots* permite hacer esta tarea rápidamente. Esta función típicamente incluye los siguientes argumentos:

outliers(x, p, tail, distribution, var.est = "mad")

donde:

- **x**: Un vector de clase **numeric** que contiene las observaciones de la variable de interés.
- **p**: probabilidad para calcular el punto de corte (entre 0 y 1). El valor por defecto de este argumento es **p = 0.99**.
- **tail**: determina si los límites se calculan para la cola superior (**"positive"**), inferior (**negative**) o ambas (**"both"**. Por defecto es "positivo", pero también puede ser "negativo" o "ambos". El valor por defecto es **tail = "positive"**.
- **distribution**: Función de distribución que se empleará para calcular el umbral para determinar los valores atípicos. Por defecto, este argumento será **distribution = "t"**. El otro posible valor es **"normal"**.

Por ejemplo, empleando la distribución t para construir tanto el umbral inferior como superior el código será

```
library(hotspots)
out_mad <- hotspots::outliers(german$monto.credito, tail = "both")
out_mad
```

```
## outlier cutoff (positive):
## [1] 6711.865
## outlier cutoff (negative):
## [1] -2072.865
```

Los resultados se pueden resumir empleando la función **summary()** de la siguiente manera:

```
summary(out_mad)
```

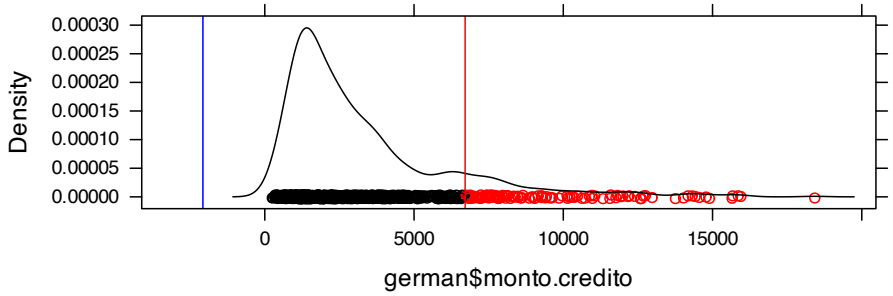
```
##
## Source data: german$monto.credito
## Distribution and probability: t, 0.99
## Tail: positive and negative outliers
## Mean:             3.271e+03
## Median:           2.320e+03
## Min:              2.500e+02
## Max:              1.842e+04
## mad:              1.627e+03
## CV (mad/median):  7.015e-01
##
## n =  1000
##
```

```
## positive outliers:
##   Cutoff  number positive outliers  % positive outliers   % sum
##     6712                       117                 11.7   34.1
##
##
## negative outliers:
##   Cutoff  number negative outliers  % negativeoutliers   % sum
##    -2073                         0                   0      0
```

Esta aproximación encuentra 117 *outliers* en la parte superior y ninguno en la parte inferior. Estos se pueden visualizar empleando la función **plot()**, de la siguiente manera:

```
plot(out_mad)
```

Figura 2.7. Observaciones atípicas (en rojo) con estimador robusto para la varianza y comparando con una distribución t (p = 0.99)



Fuente: elaboración propia.

Las observaciones atípicas las podemos obtener de manera similar a como las hemos encontrado en las subsecciones anteriores. En este caso el código sería:

```
# posición de los outliers
out_ind <- which(german$monto.credito < out_mad$negative.cut |
↪  german$monto.credito >
   out_mad$positive.cut)
out_ind
```

```
##   [1]   4   6   8  18  19  30  49  58  64  71  79  88  96 100 106 109 114 117
##  [19] 131 132 135 137 154 155 164 176 181 206 227 228 237 256 269 273 275 286
##  [37] 288 292 295 296 305 333 334 374 375 376 379 382 388 396 403 412 418 432
##  [55] 451 468 492 497 508 510 518 523 526 539 550 553 564 570 616 617 638 646
##  [73] 651 654 658 673 685 715 716 737 739 745 764 772 797 805 806 809 813 816
##  [91] 819 829 833 847 855 869 871 880 881 882 888 890 896 903 916 918 922 925
## [109] 928 940 946 954 969 972 974 981 984
```

```
# valores atípicos
out_obs <- german$monto.credito[out_ind]
```

```
out_obs
```

```
##   [1]  7882  9055  6948  8072 12579  6836  7228  9566 14421  8133  9436 12612
##  [13] 15945  7057 11938  7721  7855  7174  8487  6887 10144  8613  7758  6967
##  [25]  7308  7485  9572 10623 10961  7865 14555  7418  8978 12169 11998 10722
##  [37]  7582  9398  7629  9960 10127  7408 11590 13756 14782  7685 14318 12976
##  [49]  7374 11760  8648  7253  8471 11328 11054  7238  8318  9034  6850  8588
##  [61]  7127  7119  7966  7763  8858  6999 12389  6758 12204  9157 15653  7980
##  [73]  7476  8086 10222 10366  9857 14027  7596 11560  6761 14179 12680  8065
##  [85]  7511  7472  9271  9283  9629  7432 15857  8335 11816  6761 10875  7409
##  [97]  7678  6742  7814  9277 15672  7824  8947 10477 18424 14896 12749  6872
## [109] 10297  6842  8358 10974  7166  7393  7297  8386  8229
```

Antes de continuar nuestro estudio, es importante mencionar que el paquete *univOutl* (D'Orazio, 2022), provee una "infraestructura" que permite implementar casi todos los métodos estudiados en este capítulo. Por ejemplo, la función **boxB()** permite implementar el método del rango intercuartílico, sin y con el ajuste por distribuciones asimétricas de Hubert y Vandervieren (2008). La función típicamente incluye los siguientes argumentos:

boxB(x, k, method)

donde:

- **x**: Un vector de clase **numeric** que contiene las observaciones de la variable de interés.
- **k**: Constante no negativa que determina la extensión de los 'bigotes'. El valor por defecto es 1.5.
- **method**: Determina el método que se va a emplear. La opción **method="resistant"** corresponde a la aproximación tradicional y **method="adjbox"** implica emplear el *boxplot* ajustado de Hubert y Vandervieren (2008) para distribuciones sesgadas.

```
# install.packages('univOutl')
library(univOutl)
boxB(german$monto.credito, k = 1.5, method = "resistant")
```

```
## $quartiles
##     25%     50%     75%
## 1365.50 2319.50 3972.25
##
## $fences
##     lower     upper
## -2544.625  7882.375
##
## $excluded
## integer(0)
##
```

```
## $outliers
##  [1]   6  18  19  58  64  71  79  88  96 106 131 135 137 181 206 227 237 269 273
## [20] 275 286 292 296 305 334 374 375 379 382 396 403 418 432 451 492 497 510 526
## [39] 550 564 616 617 638 646 654 658 673 685 715 737 745 764 772 806 809 813 819
## [58] 829 833 855 882 888 896 903 916 918 922 928 946 954 981 984
##
## $lowOutl
## integer(0)
##
## $upOutl
##  [1]   6  18  19  58  64  71  79  88  96 106 131 135 137 181 206 227 237 269 273
## [20] 275 286 292 296 305 334 374 375 379 382 396 403 418 432 451 492 497 510 526
## [39] 550 564 616 617 638 646 654 658 673 685 715 737 745 764 772 806 809 813 819
## [58] 829 833 855 882 888 896 903 916 918 922 928 946 954 981 984
```

```
boxB(german$monto.credito, method = "adjbox")
```

```
## $quartiles
##     25%     50%     75%
## 1365.50 2319.50 3972.25
##
## $fences
##      lower      upper
##   499.6755 16097.1160
##
## $excluded
## integer(0)
##
## $outliers
##  [1]  27  28  40 112 158 178 250 310 380 459 472 494 591 722 726 751 812 916 965
##
## $lowOutl
##  [1]  27  28  40 112 158 178 250 310 380 459 472 494 591 722 726 751 812 965
##
## $upOutl
## [1] 916
```

En el mismo paquete podemos encontrar la función **LocScaleB()**, que puede identificar valores anómalos empleando el rango intercuartílico con un ajuste robusto para la varianza y media (**method='IQR'**)[17] y desviación media absoluta (MAD) sugerida por el método de Hampel (**method='MAD'**).

Veamos dos ejemplos que te permitirán entender cómo funciona esta función. El primer ejemplo emplea el método del rango intercuartílico con un ajuste robusto para

[17] Cuando se utiliza el argumento **method='IQR'** se implementa una versión robusta del método tradicional de detección de valores atípicos basada en el rango intercuartílico que estudiamos previamente. En este caso, los cuantiles son calculados mediante técnicas de *bootstrap* o el método robusto propuesto por Harrell y Davis (1982). Para mayor detalle puedes consultar la ayuda de esta función.

la varianza y media y el segundo emplea el método de Hampel.

```
LocScaleB(german$monto.credito, method = "IQR")
```

```
## $pars
##   median    scale
## 2319.500 1932.357
##
## $bounds
## lower.low  upper.up
## -3477.572  8116.572
##
## $excluded
## integer(0)
##
## $outliers
##  [1]   6  19  58  64  71  79  88  96 106 131 135 137 181 206 227 237 269 273 275
## [20] 286 292 296 305 334 374 375 379 382 396 403 418 432 451 492 497 510 550 564
## [39] 616 617 638 658 673 685 715 737 745 764 806 809 813 819 829 833 855 882 888
## [58] 896 903 916 918 922 928 946 954 981 984
##
## $lowOutl
## integer(0)
##
## $upOutl
##  [1]   6  19  58  64  71  79  88  96 106 131 135 137 181 206 227 237 269 273 275
## [20] 286 292 296 305 334 374 375 379 382 396 403 418 432 451 492 497 510 550 564
## [39] 616 617 638 658 673 685 715 737 745 764 806 809 813 819 829 833 855 882 888
## [58] 896 903 916 918 922 928 946 954 981 984
```

```
LocScaleB(german$monto.credito, method = "MAD")
```

```
## $pars
##   median    scale
## 2319.500 1627.153
##
## $bounds
## lower.low  upper.up
##  -2561.96   7200.96
##
## $excluded
## integer(0)
##
## $outliers
##   [1]   4   6  18  19  49  58  64  71  79  88  96 106 109 114 131 135 137 154
##  [19] 164 176 181 206 227 228 237 256 269 273 275 286 288 292 295 296 305 333
##  [37] 334 374 375 376 379 382 388 396 403 412 418 432 451 468 492 497 510 526
```

```
##  [55] 539 550 564 616 617 638 646 651 654 658 673 685 715 716 737 745 764 772
##  [73] 797 805 806 809 813 816 819 829 833 855 869 871 881 882 888 890 896 903
##  [91] 916 918 922 928 946 954 972 974 981 984
##
## $lowOutl
## integer(0)
##
## $upOutl
##   [1]   4   6  18  19  49  58  64  71  79  88  96 106 109 114 131 135 137 154
##  [19] 164 176 181 206 227 228 237 256 269 273 275 286 288 292 295 296 305 333
##  [37] 334 374 375 376 379 382 388 396 403 412 418 432 451 468 492 497 510 526
##  [55] 539 550 564 616 617 638 646 651 654 658 673 685 715 716 737 745 764 772
##  [73] 797 805 806 809 813 816 819 829 833 855 869 871 881 882 888 890 896 903
##  [91] 916 918 922 928 946 954 972 974 981 984
```

2.5 Comentarios finales

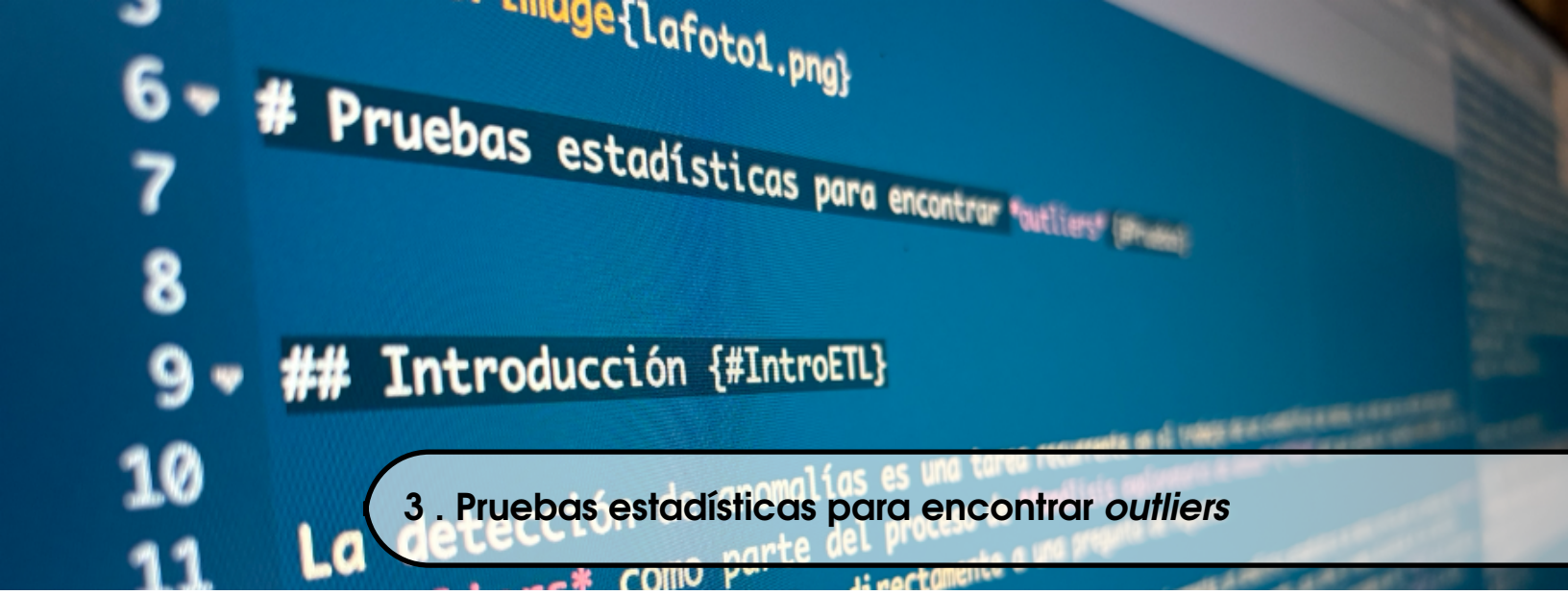
Hasta el momento hemos estudiado cómo detectar anomalías univariadas (*outliers*) empleando herramientas de la estadística que se emplean en un **EDA**. Vimos cómo emplear histogramas, el *boxplot* y ajustar el *boxplot* para la existencia de asimetría en la distribución. También discutimos técnicas que implican construir umbrales para determinar qué observaciones son atípicas como:

- Uso de percentiles
- Rango intecuartílico (prueba de Tukey)
- Método de Hampel

Así mismo, discutimos una aproximación que implica reescalar los datos para compararlos con umbrales de la distribución normal (z-score) y otra que implica emplear estimadores robustos a la presencia de valores atípicos para la varianza y media y posteriormente comparar los datos reescalados con una distribución.

Es importante anotar que solamente realizamos el análisis de una de las variables cuantitativas de la base de datos. Un **EDA** completo y un análisis de anomalías univariado completo implicarán replicar nuestro análisis para las otras variables cuantitativas. ¡Inténtalo!

La combinación de la aproximación gráfica y con métricas nos permite empezar a llenar nuestra caja de herramientas con técnicas estadísticas para detectar *outliers* (anomalías univariadas). En el Capítulo 3 estudiaremos pruebas estadísticas para detectar anomalías univariadas. En los Capítulo 4 y 5 estudiaremos aproximaciones estadísticas para detectar anomalías multivariadas. En estos capítulos seguiremos llenando nuestra caja de herramientas con aproximaciones para la detección de anomalías.



3 . Pruebas estadísticas para encontrar *outliers*

3.1 Introducción

La detección de anomalías es una tarea recurrente en el trabajo de un científico de datos, ya sea que se estén detectando *outliers* como parte del proceso de **análisis exploratorio de datos** (**EDA** por sus siglas en inglés) o con la finalidad de responder directamente a una pregunta de negocio.

En el Capítulo 2 discutimos que típicamente un análisis estadístico de datos inicia por el proceso del **EDA** y cómo la detección de *outliers* era importante en ese proceso[1]. Recordemos que el **EDA** es el proceso de examinar y analizar conjuntos de datos para comprender sus características, patrones y tendencias. El **EDA** es un paso importante en el *business analytics*, ya que ayuda a los científicos de datos a:

1. **Obtener una comprensión general de los datos** permitiendo a los científicos de datos familiarizarse con la estructura, el contenido y las características de los datos.
2. **Identificar patrones y tendencias** evidenciando patrones, tendencias y relaciones ocultas en los datos que podrían ser útiles para la construcción de modelos o la toma de decisiones.
3. **Detectar anomalías y errores** en los datos identificando valores atípicos, errores y problemas de calidad de los datos que podrían afectar los resultados del análisis.

Por otro lado, en el mundo del *business analytics* existe un proceso conocido como **Extracción, Transformación y Targa** (**ETL** por el término en inglés *Extract-Transform-Load*) en el que también las técnicas para encontrar anomalías son importantes. El **ETL** implica el acopio de los datos de diversas fuentes, su limpieza y preparación para su análisis posterior[2]. Los tres pasos principales del proceso **ETL** son:

[1] Recuerda, que como lo hemos discutido en los capítulos anteriores, no toda anomalía es un *outlier* pero todo *outlier* es una anomalía.

[2] El **ETL** típicamente es responsabilidad del ingeniero de datos, pero en algunas ocasiones a los científicos de datos les corresponde realizar esta tarea.

1. **Extracción:** Recuperar datos de sistemas de origen, como bases de datos, archivos o APIs.
2. **Transformación:** Limpiar, formatear y transformar los datos para que sean consistentes y adecuados para el análisis. Esto puede incluir "eliminar valores atípicos", corregir errores, estandarizar los formatos y agregar o calcular nuevas variables.
3. **Carga:** Transferir los datos transformados en un almacén de datos (bodega de datos o *Data Warehouse*) o entorno de análisis para su posterior uso.

El **ETL** y **EDA** son procesos complementarios en el trabajo de un científico de datos, si bien tienen un foco diferente en el proceso de preparar los datos. El **ETL** concentra su atención en la **limpieza y transformación** de datos a gran escala para que sean consistentes y adecuados para el almacenamiento y análisis, y el **EDA** en la **exploración y comprensión** de datos preparados para identificar patrones, tendencias y anomalías. Ambos procesos tienen en común la labor de limpiar los datos, si bien el foco de la limpieza no es el mismo. En el **ETL**, la limpieza de datos se centra en garantizar que los datos sean **consistentes, completos y precisos** antes de cargarlos en un *Data Warehouse* o entorno de análisis. El **ETL** debería garantizar que los datos sean confiables y adecuados para su uso en análisis posteriores. Por otro lado, en el **EDA**, la limpieza de datos se centra en identificar **datos atípicos** que podrían afectar los resultados del análisis exploratorio. Esto ayuda a garantizar que la información obtenida del **EDA** sea precisa y confiable para el modelado y la posterior toma de decisiones.

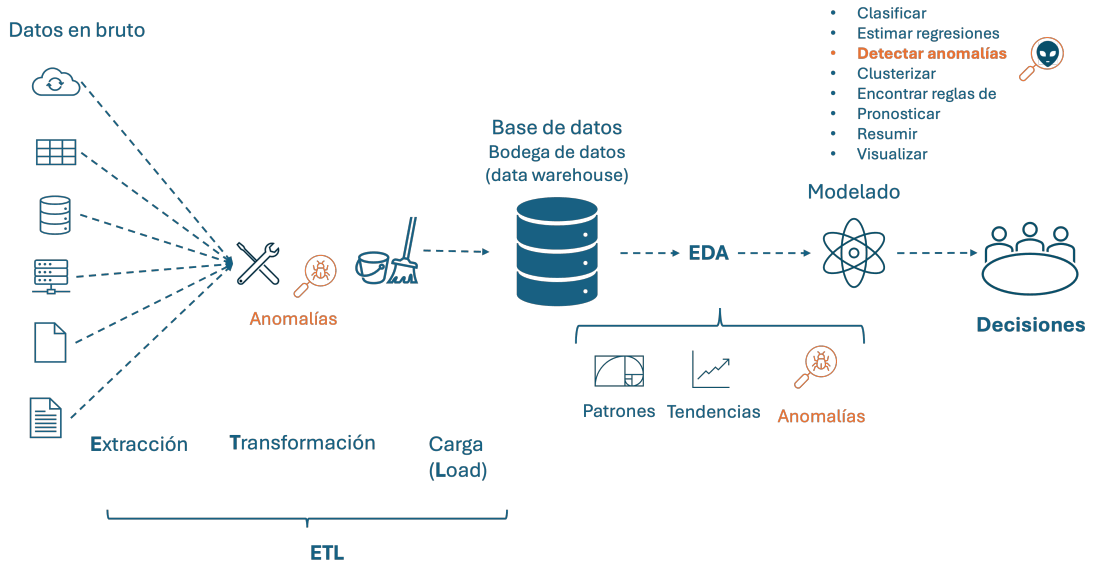
El **ETL** y **EDA** son procesos complementarios en la ciencia de datos que están relacionados (Ver Figura 3.1). La limpieza de datos es una parte crucial de **ambos** procesos, asegurando que los datos sean confiables, consistentes y adecuados para su uso en análisis y modelado. En ambos casos, la detección de anomalías hace parte del proceso de limpieza de los datos.

Como se ha mencionado, la limpieza de una base de datos en los proyectos de analítica es esencial para garantizar la calidad y confiabilidad de los resultados obtenidos. En este contexto, las técnicas de detección de *outliers* desempeñan un papel fundamental en la identificación de errores de digitación, errores de medición o individuos con comportamientos anormales, de tal manera que será necesario tomar alguna acción para garantizar que los modelos estimados o entrenados brinden los *insights* correctos para la toma de decisiones.

En la limpieza de datos, si se encuentran anomalías, pueden ser fruto de un error de registro o de medición, la decisión puede ser capturar de nuevo la medición o descartar la observación si no es posible volverla a medir o reemplazar el correspondiente valor atípico por un **NA**. Típicamente esto debería hacer parte del proceso de **ETL**. Por otro lado, la observación anómala fruto de un comportamiento diferente de un individuo debería mantenerse en la muestra y, de ser necesario, modelar ese comportamiento atípico para comprender su impacto y tomar decisiones informadas.

Finalmente, es importante recalcar que la detección de anomalías puede ser por sí misma el objetivo del ejercicio de modelado de datos del científico de datos (Ver parte superior derecha de la Figura 3.1). Típicamente las técnicas de detección de anomalías que se emplean en el **ETL** y **EDA** corresponden más a técnicas estadísticas para detectar *outliers* (anomalías puntuales o globales) univariados. Esto garantiza

Figura 3.1. La detección de anomalías en los procesos de ETL, EDA y modelado en el mundo del business analytics



Fuente: elaboración propia.

una base de datos confiable para el posterior modelado y la solución de las preguntas de negocio planteadas. Por otro lado, cuando se requiere responder preguntas de negocio que implican la tarea de detección de anomalías es más común que se empleen técnicas de detección de anomalías multivariadas, contextuales (locales) o colectivas; en especial técnicas de origen en el aprendizaje de máquina.

Como lo estudiamos en el Capítulo 1, existen diferentes tipos de anomalías y, por tanto, diferentes tipos de aproximarnos a su detección. En el Capítulo 2 estudiamos el uso de la aproximación gráfica (histogramas y *boxplots*) y el uso de métricas (percentiles, *z-score*, rango intercuartílico, método de Hampel y métricas que emplean estimadores robustos de varianza y media). Si bien estas dos aproximaciones nos dan una idea de cuáles observaciones son inusuales, no permiten realizar inferencia estadística. En otras palabras, no nos permiten tener una confianza del 95 % o 99 % de nuestras decisiones.

En este capítulo estudiaremos cuatro diferentes pruebas estadísticas que permiten hacer inferencia estadística sobre la presencia o no de datos anómalos en una variable; es decir, la existencia de *outliers* univariados. Estas cuatro pruebas hacen parte de las técnicas estadísticas formales para la identificación de valores atípicos. Las cuatro implican el cálculo de un estadístico de prueba y su comparación contra valores críticos, que depende del tamaño de la muestra, de supuestos sobre el comportamiento de la variable y del nivel de confianza o insignificancia deseado.

Las pruebas que estudiaremos aquí suponen que los datos de la variable bajo estudio, sin tener en cuenta los valores atípicos, provienen de un proceso generador de da-

tos (**DGP** por sus siglas en inglés) que sigue una distribución aproximadamente normal. Por eso es común que antes de emplear estas pruebas, el primer paso sea comprobar la normalidad de los datos utilizando análisis visual y recurriendo a pruebas de normalidad formales. No obstante, aquí aparece un problema práctico delicado, pues la existencia de valores atípicos puede hacer que las pruebas tradicionales de normalidad pierdan su poder. Es decir, en presencia de valores atípicos, las pruebas formales de normalidad tienden a rechazar la hipótesis nula de normalidad cuando en realidad los datos sí provienen de un DGP que sigue una distribución normal. Es por eso que es más común la inspección gráfica del supuesto de normalidad que el uso de las pruebas formales de normalidad.

3.2 Pruebas

En esta sección discutiremos cuatro pruebas estadísticas para determinar si existe o no *outliers* en una muestra de una variable. Para esta sección emplearemos la siguiente notación:

- X: variable cuantitativa para la que se quiere probar la presencia o no de datos atípicos.
- X_i: la i-ésima observación muestral de la variable X. Además, tendremos $i = 1, 2, \ldots, n$.
- $\bar{X}$: media muestral para todas las observaciones de la variable X.
- $\bar{X}_k$: media de la muestra que queda tras remover las k observaciones más extremas (o todas las k más pequeñas o todas las k más grandes) de la variable X.
- $\text{mín}(X)$: el valor mínimo observado para la muestra de la variable X.
- $\text{máx}(X)$: el valor máximo observado para la muestra de la variable X.
- s: desviación estándar para la muestra de la variable X.
- $X_{(i)}$: la i-esima observación tras ordenar de menor a mayor los datos. Es decir, $X_{(1)} < X_{(2)} < X_{(3)} < \ldots X_{(n-1)} < X_{(n-1)}$. Además, $X_{(1)} = \text{mín}(X)$ y $X_{(n)} = \text{máx}(X)$.

3.2.1 Prueba de Grubbs

La prueba propuesta por Grubbs (1969) es de las pruebas más empleadas para detectar un **único** *outlier* en una variable cuantitativa (X). Esta prueba permite probar las siguientes hipótesis nula (H_0) y alterna (H_A):

- H_0: La muestra proviene de una distribución normal sin *outliers*.
- H_A: Hay un *outlier* en la muestra.

Grubbs (1969) demostró que para probar esta hipótesis nula se puede emplear el siguiente estadístico de prueba:

$$G = \frac{\text{máx}\,|X_i - \bar{X}|}{s} \tag{3.1}$$

Grubbs (1969) además demostró que G sigue una distribución *t de Student* con $n-2$ grados de libertad. Nota que la anterior prueba corresponde a una prueba de dos

colas al estar buscando valores atípicos tanto en la cola inferior como en la superior de la distribución. También es posible probar si el valor mínimo (máximo) es un valor atípico. En este caso la hipótesis nula y alterna son:

- H_0: El valor mínimo (máximo) no es un *outlier*.
- H_A: El valor mínimo (máximo) es un *outlier*.

En el caso de probar si el valor mínimo es un valor atípico, el estadístico de prueba será:

$$G = \frac{\bar{X} - \text{mín}(X)}{s} \tag{3.2}$$

Y para el valor máximo tendremos:

$$G = \frac{\text{máx}(X) - \bar{X}}{s} \tag{3.3}$$

En estos dos últimos casos, los estadísticos de prueba siguen la misma distribución; la única diferencia es que se trata de pruebas de una sola cola.

La prueba de Grubbs tiene un supuesto muy fuerte: se supone que la muestra fue generada a partir de una distribución normal univariada. Este test funciona relativamente bien con muestras grandes, aunque el supuesto no se cumpla, pero puede ser muy sensible a la violación del supuesto de normalidad en muestras pequeñas. Otra desventaja de esta prueba es que está diseñada únicamente para detectar un *outlier*. Si se intuye (con el análisis gráfico o con métricas como las estudiadas en el Capítulo 2) la existencia de más de un dato atípico, esta prueba no es una buena opción.

3.2.2 Prueba de Dixon

Dixon (1950) propuso otra prueba conocida como la prueba de Dixon, la Q de Dixon (*Dixon's Q test*) o prueba Q. Esta prueba permite probar las mismas tres H_0 que la de Grubbs. Para el caso de la prueba de una cola cuya H_0 es que el valor mínimo no es un *outlier*, el estadístico de prueba propuesto por Dixon (1950) corresponde a:

$$Q_{\text{mín}} = \frac{X_{(2)} - \text{mín}(X)}{\text{máx}(X) - \text{mín}(X)} \tag{3.4}$$

Y para el caso de que la H_0 sea que el valor máximo no es un *outlier*, el estadístico de prueba propuestos por Dixon (1950) corresponde a:

$$Q_{\text{máx}} = \frac{\text{máx}(X) - X_{(n-1)}}{\text{máx}(X) - \text{mín}(X)} \tag{3.5}$$

Dixon (1950) demostró que estos estadísticos siguen una distribución especial que depende del tamaño de la muestra y del nivel de significancia. Posteriormente, Dixon (1951) provee unas tablas para tomar la decisión de esta prueba. Más adelante, Rorabacher (1991) propone una forma alternativa de encontrar los valores críticos para tomar la decisión de la prueba de Dixon. En todo caso, y al igual que la prueba de

Grubbs, la prueba depende del supuesto de que la muestra proviene de una distribución normal univariada. Esta prueba no funciona bien (ni en muestras grandes) cuando se viola el supuesto de normalidad. Además, está documentado que la prueba de Dixon es útil principalmente para muestras pequeñas ($3 \leqslant n \leqslant 30$).

3.2.3 Prueba de Rosner

La prueba de Rosner (Rosner, 1975) se emplea para detectar *outliers* en un conjunto de datos univariados, pero a diferencia de las pruebas de Grubbs y Dixon, se enfoca en detectar múltiples *outliers*.

Esta prueba tiene como hipótesis:

- H_0: La muestra proviene de una distribución normal sin *outliers*.
- H_A: Hay k *outliers* en la muestra.

La estadística de prueba para la prueba de Rosner se basa en el cálculo de las distancias máximas a la media estudentizada, que se obtienen al dividir la diferencia de cada valor extremo y la media por la desviación estándar ajustada (Ver Rosner (1975) para una descripción en detalle de esta prueba).

El estadístico de Rosner, al igual que el de Dixon, no sigue una distribución convencional y depende del número de *outliers* a probar (k). La decisión se toma comparando el estadístico calculado con los valores provistos por Rosner (1975) para diferentes niveles de significancia y tamaños de muestra.

Al igual que las pruebas anteriores, esta prueba tiene como supuesto que la muestra proviene de una distribución normal univariada. Es sensible a violaciones de la normalidad y al tamaño de la muestra. Como se mencionó anteriormente, la gran ventaja de esta prueba es que se puede emplear para identificar k valores atípicos. A diferencia de las dos pruebas anteriores (Grubbs y Dixon), para las cuales se debe recurrir a realizar el proceso de forma iterativa para verificar más de un posible atípico, si bien este procedimiento iterativo no es recomendable, en especial para la prueba de Dixon. Comúnmente, esta prueba se emplea cuando el tamaño de la muestra es grande ($n \geqslant 20$).

3.3 Empleando R para realizar pruebas de la presencia de outliers univariados

Empecemos por cargar los mismos datos del Capítulo 2. Recuerda que la base de datos correspondía a un banco comercial alemán que contenía datos de clientes a los que se les ha otorgado un crédito y se han calificado como buenos o malos deudores. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalias). La descripción de las variables se encuentra en la Sección 2.4.

Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Miremos rápidamente los datos cargados empleando la función **glimpse()** del paquete *dplyr* (Wickham et al., 2021).

```
library(dplyr)
glimpse(german)
```

```
## Rows: 1,000
## Columns: 14
## $ cuenta.corriente      <fct> less than 0 DM, 0 or less than 200 DM, no checki~
## $ historial.crediticio  <fct> critical account/ other credits existing (not at~
## $ proposito             <fct> radio/television, radio/television, education, f~
## $ monto.credito         <int> 1169, 5951, 2096, 7882, 4870, 9055, 2835, 6948, ~
## $ cuenta.ahorros        <fct> unknown/ no savings account, less than 100 DM, l~
## $ porcentaje.disponible <int> 4, 2, 2, 2, 3, 2, 3, 2, 2, 4, 3, 3, 1, 4, 2, 4, ~
## $ estadocivil.sexo      <fct> male : single, female : divorced/separated/marri~
## $ residente             <int> 4, 2, 3, 4, 4, 4, 4, 2, 4, 2, 1, 4, 1, 4, 4, 2, ~
## $ edad                  <int> 67, 22, 49, 45, 53, 35, 53, 35, 61, 28, 25, 24, ~
## $ propiedad.hogar       <fct> own, own, own, for free, for free, for free, own~
## $ numero.creditos       <int> 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, ~
## $ ocupacion             <fct> skilled employee / official, skilled employee / ~
## $ dependientes          <int> 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ trabajador.extranjero <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes~
```

Recuerda que contamos con 6 variables cuantitativas: monto.credito, porcentaje.disponible, residente, edad, numero.creditos, dependientes. En el Capítulo 2 concentramos nuestro análisis de datos atípicos en la variable `monto.credito`. Y las Figuras 2.4, 2.5 y 2.6 nos mostraban que la distribución de los datos es asimétrica. Por eso sería muy difícil asumir que los datos siguen una distribución normal como lo requieren las pruebas estudiadas anteriormente.

Indaguemos si podemos asumir que alguna de las otras 5 variables cuantitativas en nuestra base de datos sigue una distribución normal.

3.3.1 Verificación gráfica del cumplimiento del supuesto de normalidad

Una forma rápida de verificar si una muestra de datos univariados (una variable) sigue una distribución normal es empleando un diagrama qq o también conocido como un *qq plot* (*Quantile-Quantile Plot*). El diagrama qq permite comparar la distribución de un conjunto de datos con una distribución de referencia, como por ejemplo la distribución normal. En otras palabras, permite visualizar si los datos siguen una distribución específica o si se alejan significativamente de ella.

Para construir un diagrama qq, se ordenan los datos de menor a mayor y se calculan sus respectivos cuantiles (cuantiles observados). Por otro lado, se calculan los cuantiles de la distribución de referencia (por ejemplo, normal). El diagrama qq implica cons-

truir un diagrama de dispersión que tiene en el eje horizontal los cuantiles teóricos (los de la distribución de referencia) y en el eje vertical los cuantiles de la distribución de referencia.

Si los puntos del diagrama qq se encuentran aproximadamente sobre una línea recta, significa que la distribución de los datos es similar a la distribución de referencia. En cambio, si los puntos se alejan significativamente de la línea recta, indica que la distribución de los datos se diferencia de la distribución de referencia. Si se observa una curva hacia arriba, esto implica que los datos observados tienen una cola superior más pesada que la distribución de referencia. Y si se observa una curva hacia abajo, los datos tienen una cola izquierda (inferior) más pesada que la distribución de referencia. **Es importante anotar que la presencia de *outliers* puede afectar significativamente la forma del diagrama.**

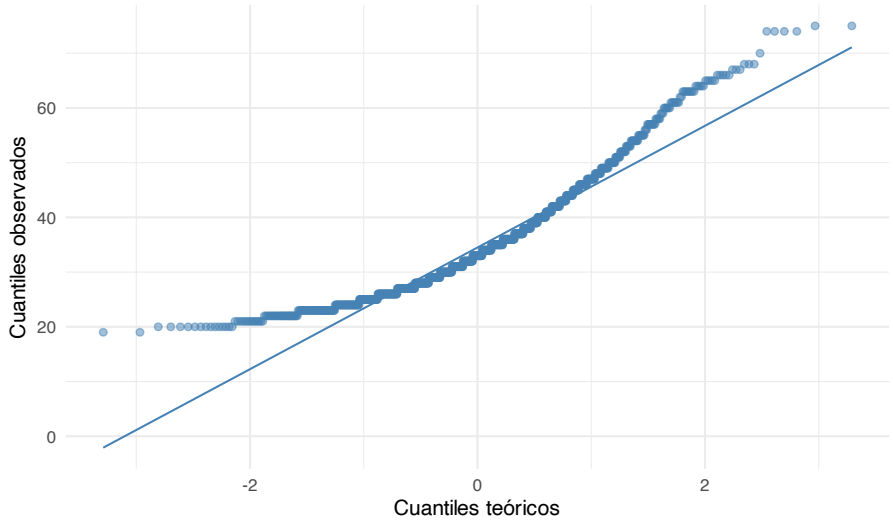
El diagrama qq se puede construir en R de muchas maneras. Por ejemplo, se puede emplear la función **qqplot()** de la base de R. También se puede emplear la función **qqPlot()** del paquete *car* (Fox y Weisberg, 2019). ¡Inténtalo! En este libro emplearemos el paquete *ggplot2* (Wickham, 2016) para construir el diagrama qq. En este paquete se cuenta con las capas **stat_qq()** y **stat_qq_line()**. La primera de estas capas grafica los puntos correspondientes al diagrama qq y la segunda traza la correspondiente línea recta. Para emplear estas dos capas, necesitaríamos emplear en la capa de estética el argumento **sample** que mapeará los datos de la variable observada al diagrama qq. Por ejemplo, para la variable `edad` el correspondiente diagrama qq (Ver Figura 3.2) se puede construir con el siguiente código:

```
library(ggplot2)

german %>%
    ggplot(aes(sample = edad)) + stat_qq(color = "steelblue", alpha = 0.5) +
    ↪  stat_qq_line(color = "steelblue") +
    xlab("Cuantiles teóricos") + ylab("Cuantiles observados") + theme_minimal()
```

Como se puede evidenciar en la Figura 3.2, las observaciones se desvían de la línea de referencia formando una curva hacia arriba. Esto es reflejo de que los datos observados tienen una cola superior más pesada que la distribución de referencia. Puedes constatar esto construyendo el respectivo histograma y *boxplot*. Es decir, la variable no evidencia seguir una distribución normal.

Te podrás estar preguntando qué tan alejados de la línea de referencia deben estar los puntos para determinar que la variable observada se distancia de la distribución normal. Para resolver esta pregunta, podemos construir intervalos de confianza para la línea de referencia, de tal manera que, si los puntos caen en el intervalo, podemos concluir que los datos no se distancian de la distribución de referencia. Esto lo podemos hacer en R empleando el paquete *qqplotr* (Almeida et al., 2018) . Este paquete provee una capa adicional que construye el intervalo de confianza para la línea de referencia, esto lo logramos con la función **stat_qq_band()**. Para poder emplear esta capa se debe reemplazar la capa **stat_qq()** por **stat_qq_point()**. El código para generar la Figura 3.3 que muestra el diagrama qq con sus respectivas bandas es el siguiente:

Figura 3.2. Diagrama qq para la variable edad



Fuente: elaboración propia.

```
# install.packages('qqplotr')
library(qqplotr)

german %>%
    ggplot(aes(sample = edad)) + stat_qq_band() + stat_qq_point(color =
    ↪  "steelblue",
    alpha = 0.5) + stat_qq_line(color = "steelblue") + xlab("Cuantiles
    ↪  teóricos") +
    ylab("Cuantiles observados") + theme_minimal()
```

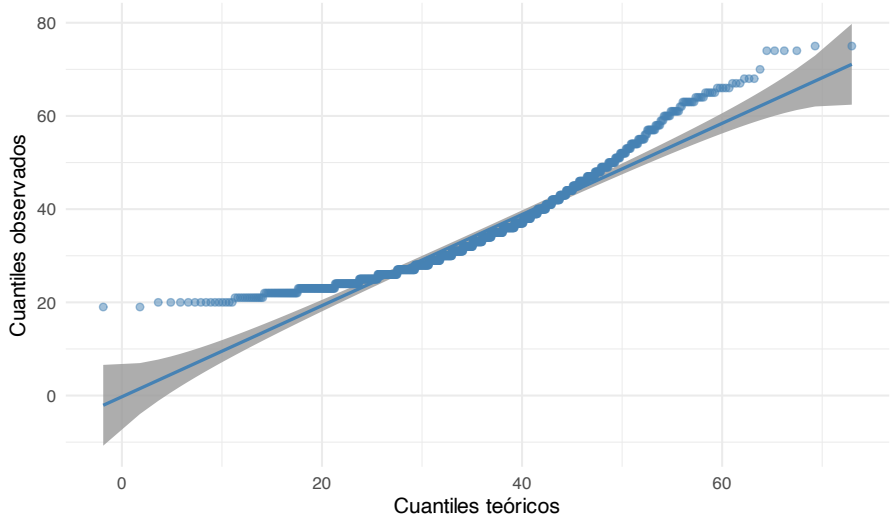
Con las bandas, la decisión es mucho más sencilla. ¡Comprueba que las variables `porcentaje.disponible`, `residente`, `numero.creditos` y `dependientes` no provienen de una distribución normal! Esto implica que las técnicas estudiadas en este capítulo no deberíamos emplearlas.

Para continuar nuestra ilustración de cómo emplear estas pruebas para detectar *outliers*, creemos dos series. Una variable `W` generada a partir de una distribución normal con un solo *outlier* y una variable `Z` con 7 *outliers*, 3 en la parte inferior y 4 en la parte superior.

Con el siguiente código podemos generar estas dos variables:

```
set.seed(1234)
datos_prueba <- data.frame(W = c(rnorm(999), 6), Z = c(rnorm(993), -9, -6, -10,
↪  10,
    8, 6, 7.5))
```

Figura 3.3. Diagrama qq con su intervalo de confianza para la variable edad



Fuente: elaboración propia.

Constatemos que las dos series generadas siguen una distribución normal. ¡Intenta reproducir los dos paneles de la Figura 3.4!

Como se puede observar en la Figura 3.4, para ambas series, la mayoría de las observaciones no se desvían de la línea de referencia (con excepción de los datos atípicos). Esto permite concluir que los datos provienen de una distribución normal. Naturalmente, en este caso sabemos que ambas variables sí fueron generadas de una distribución normal.

3.3.2 Pruebas en R

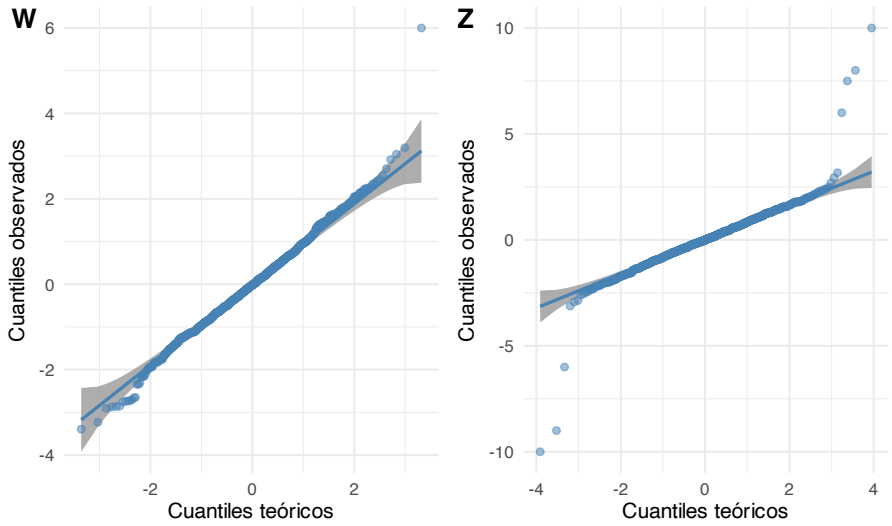
Una vez constatado que el supuesto de normalidad es plausible para la muestra, podemos proceder a realizar las pruebas para detectar la presencia de *outliers*. En esta sección estudiaremos cómo implementar las pruebas de Grubbs, Dixon y Rosner en R.

3.3.3 Prueba de Grubbs

La prueba de Grubbs se puede implementar en R por medio de la función **grubbs.test()** del paquete *outliers* (Komsta, 2022). Esta función tiene como argumento **x** los datos de la variable que deseamos chequear y el otro argumento importante de la función es **type** que especifica el tipo de prueba que se quiere realizar. Si **type = 10** se realizará una prueba para detectar si la muestra contiene un valor atípico. El lado en el que estará el valor atípico es detectado automáticamente. Si **type = 11** se comprobará si el valor más bajo y el más alto son dos valores atípicos.

Realicemos la prueba para la variable `W` que sabemos que solo tiene un valor atípico

Figura 3.4. Diagrama qq para las dos series generadas aleatoriamente



Fuente: elaboración propia.

en la parte superior. En este caso el código será:

```
# install.packages('outliers')
library(outliers)
grubbs.test(datos_prueba$W, type = 10)
```

```
##
##  Grubbs test for one outlier
##
## data:  datos_prueba$W
## G = 5.92947, U = 0.96477, p-value = 1.107e-06
## alternative hypothesis: highest value 6 is an outlier
```

El estadístico de la prueba es 5.9885, con un valor p[3] muy pequeño. Por tanto, con un nivel de confianza del 99 %, rechazamos la hipótesis de que el valor más alto 6 no es un valor atípico. En otras palabras, con base en esta prueba, concluimos que el valor más alto es un valor atípico.

En este caso, la función detectó automáticamente que se deseaba encontrar si el máximo era un outlier. Si queremos que la función pruebe si el mínimo es un *outlier*, podemos emplear el argumento **opposite = FALSE**. Este argumento le dice a la función si deseamos comprobar si el valor con mayor diferencia respecto a la media es *outlier*

[3]Como ocurre con cualquier prueba estadística, para rechazar o no la hipótesis nula podemos comparar el estadístico de prueba con el valor de la respectiva distribución o podemos emplear el valor p. Si el valor p es menor al nivel de significancia (generalmente $\alpha = 0{,}05$ o $\alpha = 0{,}01$) entonces podemos afirmar que se rechaza la hipótesis nula con una confianza del $100 \cdot (1 - \alpha)$ %. Por el contrario, si el valor p es mayor o igual al nivel de significancia, no se cuenta con evidencia suficiente para rechazar la hipótesis nula.

(**opposite = TRUE**) o el lado opuesto.

Por ejemplo, para probar si el mínimo es un *outlier* el código sería:

```
grubbs.test(datos_prueba$W, type = 10, opposite = TRUE)
```

```
##
##  Grubbs test for one outlier
##
## data:  datos_prueba$W
## G = 3.32506, U = 0.98892, p-value = 0.4288
## alternative hypothesis: lowest value -3.39606353457436 is an outlier
```

El valor p es mayor que 0.05. Con un nivel de confianza del 95 %, no rechazamos la hipótesis nula, la cual plantea que el valor más bajo no es un valor atípico. En otras palabras, no podemos concluir que el valor más bajo sea un valor atípico de acuerdo con el conjunto de datos con el que estamos trabajando.

Finalmente, si queremos detectar si el valor mínimo y máximo son valores atípicos tendríamos el siguiente código:

```
grubbs.test(datos_prueba$W, type = 11)
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  datos_prueba$W
## G = 9.25453, U = 0.95373, p-value < 2.2e-16
## alternative hypothesis: -3.39606353457436 and 6 are outliers
```

La prueba tiene un valor p aproximadamente de cero. En este caso podemos rechazar la hipótesis nula porque es inferior al nivel de significancia ($\alpha = 0{,}01$), por lo que concluimos que tenemos evidencia suficiente para determinar que los valores mínimo y máximo son valores atípicos (o solo uno de ellos) en la variable `W`.

3.3.4 Prueba de Dixon

La prueba de Dixon puede ser implementada empleando la función **dixon.test.test()** del paquete *outliers* (Komsta, 2022). Esta prueba funciona relativamente bien en muestras pequeñas, por eso la misma función solo permite muestras pequeñas entre 3 y 25 observaciones ($3 \leqslant n \leqslant 25$). Para seguir con nuestro ejemplo, utilizaremos un subconjunto de los datos simulados que tendrá 24 observaciones y el valor atípico.

```
datos_test_dixon <- c(datos_prueba$W[1:23], max(datos_prueba$W))
```

Si queremos probar la H_0 : de que el máximo valor *no* es una atípico, el código será el siguiente:

```
dixon.test(datos_test_dixon)
```

```
##
```

```
##  Dixon test for outliers
##
## data:  datos_test_dixon
## Q = 0.70238, p-value < 2.2e-16
## alternative hypothesis: highest value 6 is an outlier
```

La prueba tiene un valor p muy pequeño; dado que este valor es inferior al nivel de significancia, tenemos evidencia suficiente para rechazar la hipótesis nula (con un 99 % de confianza) y determinar que el valor máximo es un valor atípico. De la misma manera que la función **grubbs.test()**, esta función detecta automáticamente dónde está la observación con la mayor dispersión con respecto a la media. Si se desea probar (con estos datos) la H_0 de que el mínimo valor *no* es una atípico, se puede emplear el siguiente código:

```
dixon.test(datos_test_dixon, opposite = TRUE)
```

```
##
##  Dixon test for outliers
##
## data:  datos_test_dixon
## Q = 0.39279, p-value = 0.1379
## alternative hypothesis: lowest value -2.34569770262935 is an outlier
```

Dado el valor p obtenido, y aún empleando un nivel de significancia del 10 %, no podemos rechazar la hipótesis nula. La cual implica que el valor más bajo no es un valor atípico. En otras palabras, no podemos concluir que el valor mínimo sea un valor atípico de acuerdo al conjunto de datos con el que estamos trabajando.

3.3.5 Prueba de Rosner

Por otro lado, la prueba de Rosner se emplea cuando el tamaño de la muestra es superior a 20 ($n \geqslant 20$). Esta se puede implementar con la función **rosnerTest()** del paquete *EnvStats* (Millard, 2013). Esta función requiere los siguientes tres argumentos:

rosnerTest(x, k, alpha)

donde:

- **x**: vector numérico de observaciones.
- **k**: número entero positivo que indica el número de observaciones atípicas sospechosas. El argumento **k** debe estar comprendido entre 1 y $n - 2$. El valor por defecto es **k = 3**.
- **alpha**: número que puede tomar valores entre 0 y 1 y que corresponde al error tipo I asociado a la prueba (como predeterminado **alpha = 0.05**).

Por ejemplo, para la serie z probemos la existencia de 10 valores atípicos, recuerden que generamos solo 7 *outliers*. Eso lo podemos realizar con el siguiente código.

```
# Cargar el paquete
library(EnvStats)
```

```r
# Realizar la prueba de Rosner
rst <- rosnerTest(datos_prueba$Z, k = 10, alpha = 0.01)

print(rst)
```

```
##
## Results of Outlier Test
## -------------------------
##
## Test Method:                     Rosner's Test for Outliers
##
## Hypothesized Distribution:       Normal
##
## Data:                            datos_prueba$Z
##
## Sample Size:                     1000
##
## Test Statistics:                 R.1  = 8.385497
##                                  R.2  = 8.650771
##                                  R.3  = 8.132802
##                                  R.4  = 7.435410
##                                  R.5  = 7.175184
##                                  R.6  = 5.920776
##                                  R.7  = 5.993057
##                                  R.8  = 3.218524
##                                  R.9  = 3.209715
##                                  R.10 = 3.031358
##
## Test Statistic Parameter:        k = 10
##
## Alternative Hypothesis:          Up to 10 observations are not
##                                  from the same Distribution.
##
## Type I Error:                    1%
##
## Number of Outliers Detected:     7
##
##    i      Mean.i      SD.i      Value Obs.Num    R.i+1 lambda.i+1 Outlier
## 1  0 0.019401810 1.1948490 -10.000000     996 8.385497   4.396763    TRUE
## 2  1 0.029431241 1.1525642  10.000000     997 8.650771   4.396529    TRUE
## 3  2 0.019440691 1.1090200  -9.000000     994 8.132802   4.396295    TRUE
## 4  3 0.028487272 1.0721014   8.000000     998 7.435410   4.396061    TRUE
## 5  4 0.020483745 1.0424145   7.500000    1000 7.175184   4.395826    TRUE
## 6  5 0.012966643 1.0155707  -6.000000     995 5.920776   4.395592    TRUE
## 7  6 0.019015905 0.9979856   6.000000     999 5.993057   4.395357    TRUE
## 8  7 0.012992759 0.9802460   3.167938     816 3.218524   4.395122   FALSE
```

```
## 9  8 0.009812371 0.9756012  -3.121590     661 3.209715   4.394886   FALSE
## 10 9 0.012972212 0.9710017  -2.930482     122 3.031358   4.394650   FALSE
```

Como se puede evidenciar en los resultados, vemos que existen 7 valores atípicos con un nivel de confianza del 99%, observaciones que corresponden a aquellas que generamos.

3.4 Comentarios finales

En este capítulo continuamos el análisis de anomalías univariadas. Incorporamos en nuestra caja de herramientas 3 pruebas estadísticas para detectar la presencia de *ouliers* (anomalías puntuales). Las pruebas estadísticas permiten una decisión formal sobre la presencia de *outliers*, pero deben usarse con cautela. Sus resultados dependen de los supuestos de normalidad y del tamaño de muestra, por lo que conviene complementarlas con la validación de los supuestos y cuando sea pertinente con las herramientas gráficas y métricas descriptivas presentadas en el Capítulo 2.

Antes de continuar con el estudio de técnicas para detectar anomalías multivariadas en el Capítulo 4, es importante resaltar que cuando se realiza una prueba de valores atípicos se puede caer en lo que se conoce en la jerga de la literatura de detección de anomalías como enmascaramiento (*masking*) y *swamping*[4].

Para entender estos dos conceptos reconozcamos que cuando se realiza una prueba de valores atípicos, es necesario elegir un procedimiento basado en el número de valores atípicos (como por ejemplo las pruebas de Grubbs y Dixon) o especificar el número de valores atípicos para una prueba (como la prueba de Rosner). Las pruebas de Grubbs y Dixon solo comprueban un valor atípico. Sin embargo, otros procedimientos, como la prueba de Rosner y la de Tietjen-Moore (Tietjen y Moore, 1972)[5], requieren que se especifique el número de valores atípicos. Es difícil hacerlo correctamente. Al fin y al cabo, estamos realizando la prueba para encontrar valores atípicos. El enmascaramiento y el *swamping* son dos problemas que pueden producirse cuando se especifica un número incorrecto de valores atípicos en un conjunto de datos.

El enmascaramiento se produce cuando se especifican muy pocos valores atípicos. En general, se dice que un valor atípico enmascara un segundo valor atípico, si el segundo valor atípico puede considerarse un valor atípico solo por sí mismo, pero no en presencia del primer valor atípico. Así, tras la eliminación del primer valor atípico, el segundo aparece como tal. Los valores atípicos adicionales que existan pueden afectar a la prueba para que no detecte ningún valor atípico. Por ejemplo, si se especifica un valor atípico cuando hay dos, la prueba puede pasar por alto ambos valores atípicos.

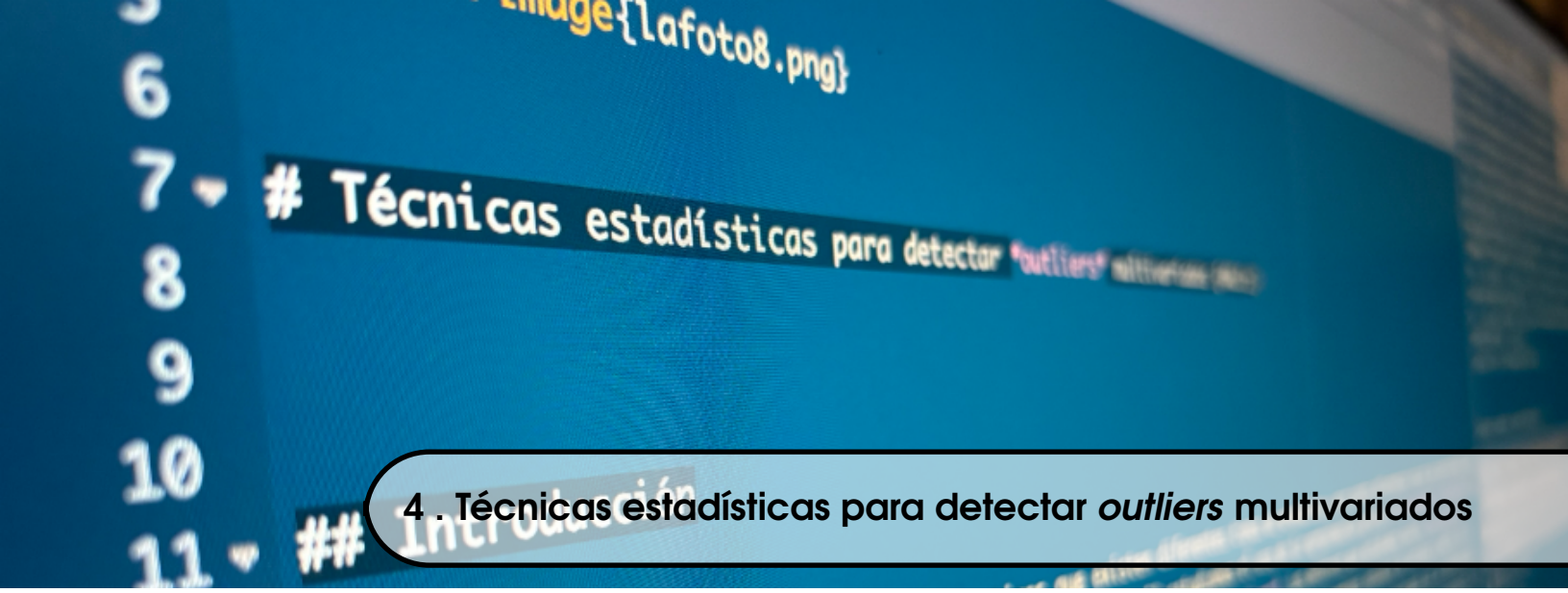
Por el contrario, cuando se especifica un número excesivo de valores atípicos, se produce un efecto *swamping*. Se dice que un valor atípico "empantana" una segunda observación, si esta última solo puede considerarse un valor atípico en presencia del primero. En este caso, la prueba identifica demasiados puntos de datos como valores

[4] La traducción sería "empantanamiento" o "embarrada". Un *swamp* es un pantano.
[5] Esta prueba no la estudiamos en este capítulo.

atípicos. Por ejemplo, si especifica dos valores atípicos cuando solo hay uno, la prueba podría determinar que hay dos valores atípicos.

Así, cuando estamos empleando técnicas de detección de anomalías, como las pruebas de Grubbs y Dixon, es importante tener en cuenta la posibilidad del *masking* y *swamping*.

En el Capítulo 4 continuaremos nuestro estudio de herramientas para la detección de anomalías concentrándonos en técnicas para detectar *outliers* multivariados. Por ahora recuerda que las anomalías pueden ser de diferentes tipos y por eso es importante que tengamos diversas herramientas para su detección.



4 . Técnicas estadísticas para detectar *outliers* multivariados

4.1 Introducción

En el Capítulo 1, discutimos que existen diferentes tipos de anomalías y por tanto diferentes tipos de aproximaciones para su detección. En el Capítulo 2 estudiamos el uso de la aproximación gráfica (*histogramas* y *boxplots*) y el uso de métricas para "marcar" observaciones como posibles *outliers*. Las métricas que estudiamos fueron: *percentiles*, *z-score*, *rango intercuartílico*, *método de Hampel* y métricas que emplean estimadores robustos de varianza y media. En el Capítulo 3 estudiamos métodos de origen en la estadística para detectar anomalías univariadas con un determinado nivel de confianza, específicamente cuatro diferentes pruebas que permiten hacer inferencia estadística sobre la presencia o no de datos atípicos (*outliers*) en una variable; es decir, la existencia de anomalías univariadas[1].

En este capítulo discutimos aproximaciones desde la estadística para detectar *outliers* multivariados. Así como en el caso de las anomalías univariadas, la estadística ha desarrollado métricas y pruebas para detectar *outliers* multivariados. En este capítulo y en este libro solo cubriremos las aproximaciones estadísticas que implican el cálculo de una métrica de distancia y la comparación de las métricas de las observaciones con un umbral para ser consideradas como valores atípicos. Más adelante en el Capítulo 5 estudiaremos una técnica matemática de reducción de dimensionalidad que es empleada en la estadística para la detección de *outliers*. Y posteriormente, en los capítulos 8 a 10 estudiaremos técnicas del aprendizaje de máquina que sirven para encontrar anomalías multivariadas.

4.2 Métricas basadas en distancias

Las medidas de distancia (o similitudes) nos permiten cuantificar la proximidad entre dos objetos estadísticos. Estos objetos pueden ser dos variables aleatorias, dos distribu-

[1] Recuerda, que como lo hemos discutido en los capítulos anteriores, no toda anomalía es un *outlier* pero todo *outlier* es una anomalía.

ciones de probabilidad o una observación frente a otras observaciones. Las distancias se emplean en muchas tareas de la analítica como en la tarea de clústering en que las medidas de similitud juegan un papel importante en determinar qué observaciones se parecen entre si y cuáles no[2] y la tarea de clasificación en modelos como el **kNN** (k vecinos próximos) donde las medidas de distancia permiten escoger los vecinos más próximos[3].

En la tarea de detección de anomalías, las medidas de distancia son una herramienta natural para medir qué tan cerca se encuentra una observación de otra. En especial son útiles para medir la cercanía entre observaciones cuando tenemos múltiples variables (*features*). Existen muchas medidas de distancia como la distancia euclidiana, la distancia de Chebyshev (también conocida como la distancia máxima), la distancia de Manhattan, la distancia de Canberra, la distancia de Minkowski y la distancia binaria (Ver sección 8.1 para una breve discusión de estas distancias[4]).

De todas las distancias, la más conocida y empleada es la distancia euclidiana. Desde el punto de vista geométrico, la distancia euclidiana entre dos observaciones es la distancia más corta posible entre ellas.

Formalmente, definamos el vector de las observaciones para las d variables para un individuo i, como $\mathbf{x_i} = [x_{i,1} \quad x_{2,1} \quad \cdots \quad x_{i,d}]$; en otras palabras, la i-ésima fila de la matriz de datos $\mathbf{X}_{n\times d}$. Entonces, la **distancia euclidiana** se define como:

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\left(\sum_{m=1}^{d} \left(x_{i,m} - x_{j,m}\right)^2\right)} = \sqrt{\left(\mathbf{x_i} - \mathbf{x_j}\right)^T \left(\mathbf{x_i} - \mathbf{x_j}\right)} \tag{4.1}$$

Nota que para cada observación se puede calcular la distancia con respecto a las otras $n-1$ observaciones, de tal manera que terminamos con $(n(n-1))/2$ pares de distancias.

4.2.1 Distancia de Mahalanobis

Un problema de la medida de distancia euclidiana es que no tiene en cuenta la correlación entre las variables. En caso de existir correlación entre las variables, la distancia euclidiana asigna el mismo peso a dichas variables y, puesto que estas variables miden esencialmente la misma característica, esta única característica recibe un peso adicional (Mclachlan, 1999).

Mahalanobis (1936) propuso una medida de distancia que resolviera este problema. La medida propuesta por el autor terminó tomando su apellido y hoy se conoce como la **distancia Mahalanobis**. La idea en esta distancia es escalar la contribución de las variables individuales con el valor de la distancia en función de la variabilidad

[2] En la Sección 2.2 de Alonso et al. (2025) se presenta una discusión de las diferentes distancias y cómo se emplean estas en los algoritmos de clústering.

[3] En el Capítulo 6 de Alonso y Hoyos (2025) se presenta una discusión de como se emplean las diferentes distancias en el algoritmo **kNN**.

[4] En la Sección 2.2 de Alonso et al. (2025) puedes consultar una presentación más detallada de estas distancias.

de cada variable[5]. A diferencia de las distancias más conocidas como la euclidiana, en la que se asume independencia entre las variables, la **distancia Mahalanobis** usa la correlación entre las variables para ajustar su separación; este aspecto la hace muy útil en el caso en que el conjunto de datos tiene correlaciones entre sí.

La distancia de Mahalanobis se calcula para cada observación con respecto al centro de la distribución de datos; es decir, la media multivariada del conjunto de datos. De esta manera tendremos una distancia de Mahalanobis por cada observación. A diferencia de la distancia euclidiana que se calcula entre las parejas de observaciones.

Formalmente, la **distancia de Mahalanobis** para la i-ésima observación $MD(\mathbf{x_i})$ se define como:

$$MD(\mathbf{x_i}) = \sqrt{(\mathbf{x_i} - \bar{\mathbf{X}})^T \Sigma^{-1} (\mathbf{x_i} - \bar{\mathbf{X}})} \tag{4.2}$$

donde $\bar{\mathbf{X}}$ corresponde al vector que contiene las medias para cada variable; es decir, la media de cada columna de la matriz $\mathbf{X}_{n \times d}$. Y finalmente,

$$\Sigma = \begin{bmatrix} S_1^2 & S_{1,2} & \cdots & S_{1,d} \\ S_{1,2} & S_2^2 & \cdots & S_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ S_{1,d} & S_{2,d} & \cdots & S_d^2 \end{bmatrix} \tag{4.3}$$

donde $S_{k,l}$ es la covarianza entre la k-ésima variable (columna k de la matriz $\mathbf{X}_{n \times p}$) y la l-ésima variable y S_k^2 es la varianza de la k-ésima variable. No sobra enfatizar que mientras que la distancia euclidiana mide la distancia directa entre dos observaciones, la **distancia de Mahalanobis** mide qué tan lejos está una observación del centro de su distribución (multivariada), ajustando por la varianza y la covarianza de los datos.

Nota que en últimas, **la distancia de Mahalanobis mide el número de desviaciones estándar que hay entre una observación y la media de una distribución**. Por esta razón, la **distancia Mahalanobis** es muy empleada para la detección de múltiples valores atípicos (*outliers*) en variables cuantitativas.

Adicionalmente, se ha documentado que la distancia de Mahalanobis es menos sensible a la presencia de *outliers* en el conjunto de datos. Por otro lado, el cálculo de la distancia de Mahalanobis requiere la inversión de la matriz de varianzas y covarianza, lo que puede ser computacionalmente costoso para conjuntos de datos grandes.

Para identificar los valores atípicos, la **distancia de Mahalanobis** entre cada observación i y el centro en los datos de n dimensiones se necesita fijar un umbral. Algunos autores suelen fijarse arbitrariamente el umbral en 3 desviaciones estándar. Sin embargo, autores como Rousseeuw y Van Zomeren (1990) han demostrado que la **distancia de Mahalanobis** puede aproximarse mediante una distribución Chi-cuadrado (χ^2) con d (el número de variables o *features*) grados de libertad. De esta manera, las observaciones con una **distancia de Mahalanobis** ($MD(\mathbf{x_i})$) mayor que $\chi^2_{d,\alpha}$ serán marcadas como *outliers* por esta aproximación. Por otro lado, algunos autores como Cabana

[5]Por eso esta distancia también es conocida como la **distancia estadística** o **distancia ponderada**.

et al. (2021) sugieren emplear α en 0,025; es decir, emplear el umbral en el 2,5 % de observaciones más extremas.

4.2.2 Distancia robusta de Mahalanobis

En la Sección 2.3.5 discutimos cómo los estimadores tanto de la media como de la desviación estándar pueden tener problemas de sesgo en la presencia de *outliers*. La situación que observamos en el análisis univariado también está presente en el caso multivariado; y ahora se le suma el sesgo posible en la covarianza y por tanto en la correlación. Una solución natural es emplear estimadores robustos a la presencia de valores atípicos para resolver el problema, tal como lo realizamos en el caso univariado.

Autores como Rousseeuw y Zomeren (1990) propusieron emplear una *distancia robusta de Mahalanobis*. Esta propuesta implica sustituir los estimadores de los parámetros de la media ($\bar{\mathbf{X}}$) y la matriz de varianzas y covarianzas (Σ). Por ejemplo, Rousseeuw y Zomeren (1990) sugieren estimar esos parámetros a través del método *elipsoide de volumen mínimo* (**MVE** por las siglas en inglés). Otros autores como EStimator (1999) y Rousseeuw (1984) sugieren un método alternativo conocido como **MCD** (*Minimum Covariance Determinant*).

En general la **distancia robusta Mahalanobis** (**RMD**) viene dada por la siguiente expresión:

$$RMD(\mathbf{x_i}) = \sqrt{(\mathbf{x_i} - \bar{\mathbf{X}}_R)^T {\Sigma_R}^{-1} (\mathbf{x_i} - \bar{\mathbf{X}}_R)} \tag{4.4}$$

donde $\bar{\mathbf{X}}_R)$ y Σ_R son los estimadores robustos para media y matriz de covarianza. Por ejemplo, se podría emplear el estimador robusto **MVE** o el **MCD** u otro. De igual manera a lo realizado anteriormente, la $RMD(\mathbf{x_i})$ se comparará con el umbral que provee la distribución χ^2 con d grados de libertad.

Leys et al. (2018) sostienen que la distancia de Mahalanobis no es una buena aproximación para identificar valores atípicos, ya que utiliza las medias y las covarianzas de todos los datos (incluidos los *outliers*) para determinar las distancias individuales. Rousseeuw (1984) sugiere calcular la matriz robusta de covarianza mediante el uso del *Determinante Mínimo de Covarianza* (*MCD* por la sigla del término en inglés *Minimum Covariance Determinant*). La idea detrás de *MCD* es encontrar observaciones cuya covarianza tenga el menor determinante, produciendo un subconjunto de observaciones a partir del cual calcular las estimaciones de la media y covarianza.

El **MCD** calcula la media y la matriz de varianzas y covarianza empleando el subconjunto más central de datos (típicamente dos tercios de los datos), antes de calcular la distancia de Mahalanobis. Se considera que este es un método más sólido para identificar y eliminar valores atípicos que la distancia de Mahalanobis original.

El **MCD** implica los siguientes pasos:

- Buscar aquellas h observaciones cuya matriz de covarianza tiene el determinante más bajo posible.
- Calcular la media de estas h observaciones.

- Calcular la matriz de varianzas y covarianza de muestra de estas h observaciones (multiplicada por el factor de consistencia).
- Calcular la **distancia de Mahalanobis** con estos nuevos estimadores de la media y la matriz de varianzas y covarianzas y la expresión (4.4).

El cálculo de **MCD** es computacionalmente demandante, pero existen algunos algoritmos que pueden agilizar el cálculo.

4.3 Implementación en R

Para realizar un ejemplo práctico seguiremos usando los mismos datos que empleamos en los Capítulos 2 y 3. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalÃŋas). La descripción de las variables se encuentra en la Sección 2.4.

Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Seleccionemos solo las variables cuantitativas para poder aplicar las técnicas estudiadas en las secciones anteriores. Empleemos la función **select_if()** del paquete *dplyr* (Wickham et al., 2021) para hacer esta tarea más sencilla.

```
library(dplyr)
german_cuanti <- german %>%
    select_if(is.numeric)
```

Noten que contamos con una matriz $\mathbf{X}_{n\times d}$ de dimensiones 1000×6 (mil observaciones y 6 variables).

4.3.1 Distancia de Mahalanobis

La **distancia de Mahalanobis** se puede implementar en R por medio de la función **mahalanobis()** del paquete *stats* (R Core Team, 2013). Esta función calcula las distancias entre cada observación y el punto central de los datos empleando tres argumentos: **x**, **center** y **cov**. Similar a lo ya estudiado, el argumento **x** corresponde a los datos multivariados (de clase **matrix** o **data.frame**), **center** es el vector de medias de las variables y **cov** es la matriz de varianzas y covarianza de los datos.

La **distancia de Mahalanobis** para cada una de las observaciones teniendo en cuenta todas las variables cuantitativas se puede calcular con la siguiente línea de código:

```
# Encontrar las distancias de Mahalanobis para cada observación
MD <- mahalanobis(x = german_cuanti, center = colMeans(german_cuanti), cov =
↪  cov(german_cuanti))
```

En la anterior línea de código hemos empleado las funciones **colMeans()** y **cov()** de la base de R para calcular el vector de medias ($\bar{\mathbf{X}}$) y la matriz de varianzas y covarianzas (Σ) de los datos, respectivamente.

Para determinar qué observaciones se pueden considerar *outliers*, comparemos la ($MD(\mathbf{x_i})$) con $\chi^2_{6,0,025}$, tal como lo sugieren Cabana et al. (2021). El umbral lo podemos construir empleando la función **qchisq()** de la base de R. El código que genera el umbral será:

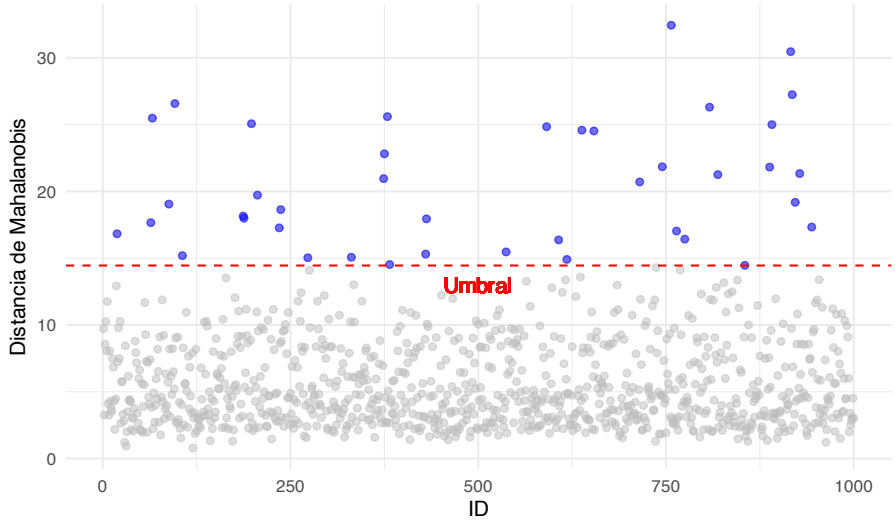
```
# Valor de corte p = 0.95 df = 6 con ncol(german_cuanti)
umbral <- qchisq(p = 0.975, df = ncol(german_cuanti))
```

Finalmente, para determinar que observaciones son *outliers*, podemos emplear el siguiente código:

```
## Obtener las observaciones superiores al valor de corte
anomalias_MD <- german_cuanti[MD > umbral, ]
```

Con este método hemos detectado 41 *outliers*. Una forma de visualizar estas observaciones es emplear un gráfico como el que se presenta en la Figura 4.1.

Figura 4.1. Clientes clasificados como outliers según la distancia de Mahalanobis



Fuente: elaboración propia.

La Figura 4.1 fue creada empleando el paquete *ggplot2* y la función **gghighlight()** del paquete *gghighlight* (Yutani, 2022). El código empleado fue el siguiente:

```
# Crear datos para la visaulización
MD_datos <- data.frame(ID = 1:1000, MD = MD)

library(ggplot2)
```

```
library(gghighlight)

# Construir la visualización
MD_datos %>%
    ggplot(aes(x = ID, y = MD)) + geom_point(color = "blue", alpha = 0.5) +
    ↪  ylab("Distancia de Mahalanobis") +
    theme_minimal() + gghighlight(MD > umbral, use_direct_label = F) +
    ↪  geom_hline(yintercept = umbral,
    linetype = "dashed", color = "red") + geom_text(x = 500, y = 13, label =
    ↪  "Umbral",
    size = 4, color = "red")
```

4.3.2 Distancia robusta de Mahalanobis

La *distancia robusta de Mahalanobis* se puede obtener en R recurriendo a la función **robustMD()** del paquete *faoutlier* (Chalmers y Flora, 2015). Recuerden que esta función solo es posible aplicarla a variables continuas.

Los argumentos de esta función son los datos multivariados que pueden ser de clase **matrix** o **data.frame** (**x**) y el método robusto de estimación de los parámetros de la media ($\bar{\mathbf{X}}$) y la matriz de varianzas y covarianzas (Σ) (**method**). Si empleamos **method = "mve"**, se estimarán estos parámetros con el método *elipsoide de volumen mínimo* (**MVE**) sugerido por Rousseeuw y Zomeren (1990). Si **method = "mcd"** se empleará el método *MCD* sugerido por autores como EStimator (1999) y Rousseeuw (1984).

Ahora calculemos primero la *distancia robusta de Mahalanobis* empleando el método **MVE**. En este caso el código será:

```
# install.packages('faoutlier')
library(faoutlier)

RMD_MVE <- robustMD(german_cuanti, method = "mve")
```

Si corres este código, obtendrás un mensaje de error[6]. El error ocurre porque la variable `dependientes` solo toma dos valores (1 y 2) y por tanto el **IQR** es cero. Esto no permite el cálculo de la matriz de varianzas y covarianzas con el método *MVE*. Por eso, para emplear este método retiraremos esta variable (la sexta columna) que no genera variabilidad. En este caso el código será:

```
# install.packages('faoutlier')
library(faoutlier)

RMD_MVE <- robustMD(german_cuanti[, 1:5], method = "mve")

RMD_MVE
```

[6] El mensaje de error será algo como "*Error in cov.rob(data, method = method, ...) : at least one column has IQR 0*".

```
##           mah p  sig
## 916 146.15509 0 ****
## 96  116.24271 0 ****
## 638 107.19605 0 ****
## 819 105.74878 0 ****
## 888 102.73827 0 ****
## 918 102.23408 0 ****
## 375 100.93038 0 ****
## 379  98.65572 0 ****
## 745  92.45154 0 ****
## 715  87.81617 0 ****
```

La salida de esta función no solo muestra la *distancia robusta de Mahalanobis* sino que también la compara con el respectivo umbral y presenta el respectivo valor p. Pero tienes que tener cuidado, pues por defecto solo presenta las primeras 10 observaciones con la mayor distancia. Esto no significa que estas sean las únicas observaciones consideradas *outliers*. En el *slot* `$mah`del objeto `RMD_MVE` podemos encontrar la *distancia robusta de Mahalanobis*. De manera similar a lo que hicimos anteriormente, podemos ver cuáles son los *outliers* con el siguiente código:

```
# Valor de corte p = 0.95 df = 5 con ncol(german_cuanti)-1
umbral <- qchisq(p = 0.975, df = ncol(german_cuanti) - 1)
## Obtener las observaciones superiores al valor de corte
anomalias_RMD_MVE <- german_cuanti[RMD_MVE$mah > umbral, ]
```

En este caso obtenemos 187 *outliers*. Una forma alternativa para obtener los *outliers* es empleando el valor p con el siguiente código:

```
anomalias_RMD_MVE2 <- german_cuanti[RMD_MVE$mah_p < 0.025, ]
```

Por otro lado, el paquete *faoutlier* permite construir rápidamente una visualización similar a la Figura 4.1, pero la gráfica se generará con la base de R y no con el paquete *ggplot2*. El código que genera la Figura 4.2 es el siguiente:

```
plot(RMD_MVE, type = "xyplot", main = "", ylab = "Distancia robusta de
↪  Mahalanobis")
```

Podemos mejorar la Figura 4.2 empleando el paquete *ggplot*, obteniendo una visualización como la que se presenta en la Figura 4.3. Intencionalmente no presentamos el código que genera dicha figura, ¡intenta reproducirla!

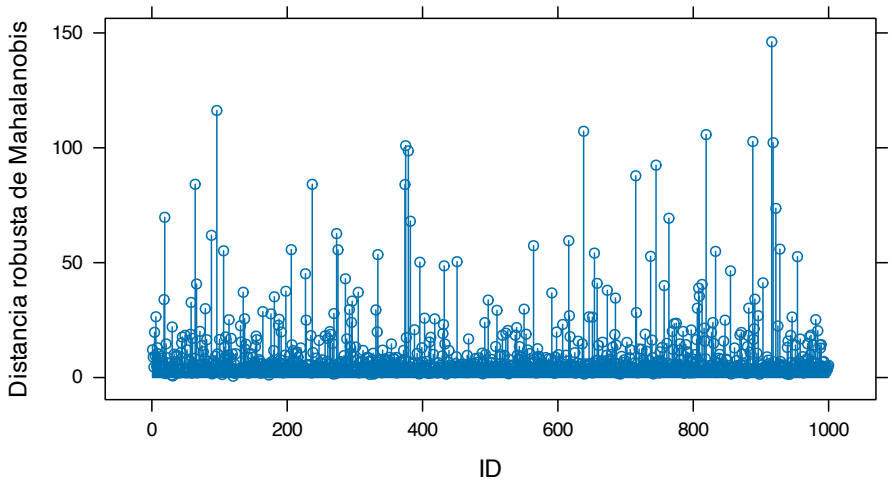
Ahora calculemos la *distancia robusta de Mahalanobis* empleando el método **MCD**. El código sería:

```
RMD_MCD <- robustMD(german_cuanti[, 1:5], method = "mcd")
```

Si corres este código, nuevamente obtendrás el mismo mensaje de error que antes[7]

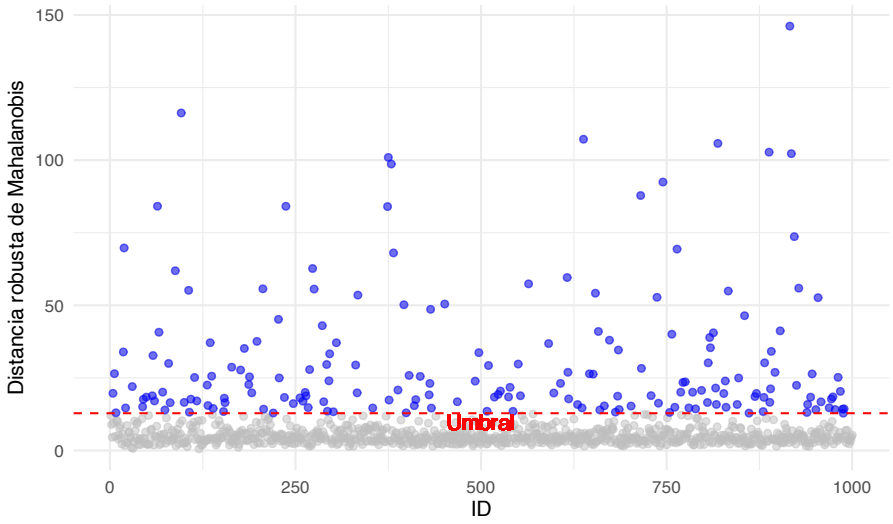
[7] El mensaje de error será algo como "*Error in cov.rob(data, method = method, ...) : at least one column has IQR 0*".

Figura 4.2. Clientes atípicos según la distancia robusta de Mahalanobis calculada con el método MVE



Fuente: elaboración propia.

Figura 4.3. Clientes atípicos según la distancia robusta de Mahalanobis calculada con el método MVE empleando ggplot2



Fuente: elaboración propia.

Eliminando la variable `dependientes` la función tampoco podrá calcular esta distancia. Ahora la razón es diferente[8], no se puede encontrar la inversa de la matriz de varianzas y covarianzas porque existen unas columnas que están muy correlacionadas (no son linealmente independientes). Si inspeccionamos con más cuidado los datos, las variables `porcentaje.disponible`, `residente` y `numero.creditos` solo toman cuatro valores (1, 2, 3 y 4) y por tanto funcionan más como variables cualitativas que cuantitativas, no presentan mucha dispersión. Por eso el cálculo de la matriz de varianzas y covarianzas con el método *MCD* no es posible. Por eso, para emplear este método retiraremos estas variables, dejando solo `monto.credito` y `edad` (la primera y cuarta columnas). En este caso el código será:

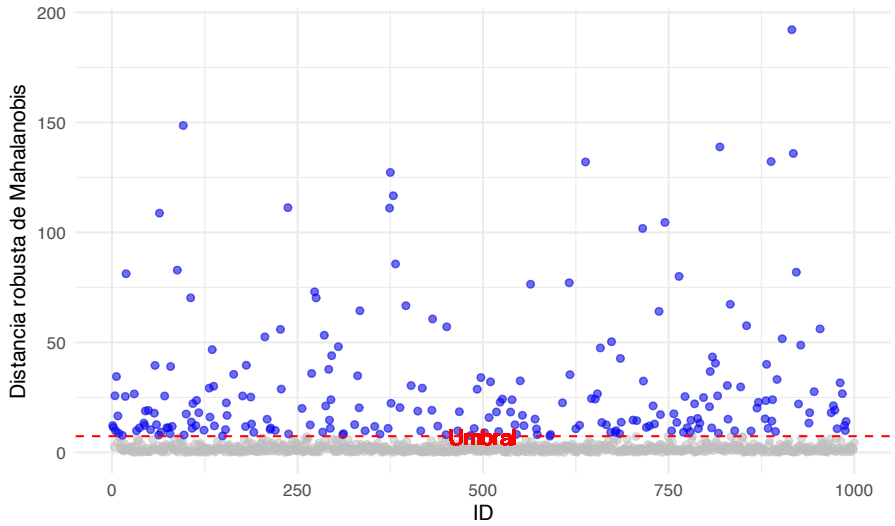
```
RMD_MCD <- robustMD(german_cuanti[, c(1, 4)], method = "mcd")
```

En este caso, los *outliers* los obtenemos (de manera análoga al caso anterior) de la siguiente manera:

```
anomalias_RMD_MCD <- german_cuanti[RMD_MCD$mah_p < 0.025, ]
```

Ahora tenemos 222 *outliers*. Y en la Figura 4.4 se visualizan estas distancias. Intencionalmente no presentamos el código que genera dicha figura, ¡intenta reproducirla!

Figura 4.4. Clientes atípicos según la distancia robusta de Mahalanobis calculada con el método MCD empleando ggplot2



Fuente: elaboración propia.

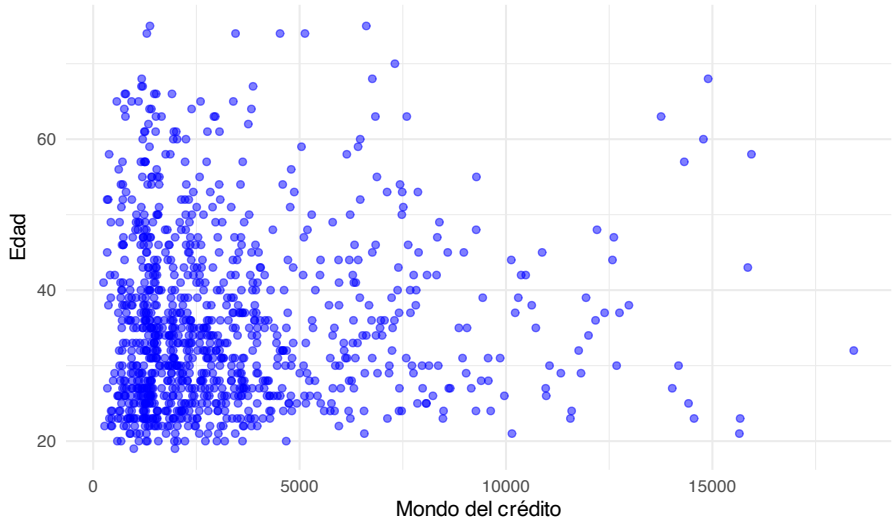
[8]El mensaje de error es "*Error in solve.default(cov, ...): Lapack routine dgesv: system is exactly singular: U[5,5] = 0*".

4.4 El caso de dos features

Cuando se cuenta con solo dos variables, es común que se emplee una aproximación gráfica para identificar los *outliers* multivariados con la *distancia de Mahalanobis*. La visualización implica tener en los ejes cada una de las variables y dibujar elípses para representar niveles de igual distancia de Mahalanobis. Cada elipse corresponde a un valor constante de distancia de Mahalanobis y el centro de las elípses corresponde a las medias de las variables. También es común no pintar todas las elípses, sino solo la que corresponde al umbral a partir del cual la *distancia de Mahalanobis* se considera muy grande y por tanto las observaciones por fuera de la elipse son etiquetadas como *outliers*.

Por ejemplo, consideremos solo las variables `monto.credito` y `edad`. El correspondiente diagrama de dispersión se presenta en la Figura 4.5. Intencionalmente no presentamos el código que genera dicha figura.

Figura 4.5. Edad y monto de crédito para los clientes del banco



Fuente: elaboración propia.

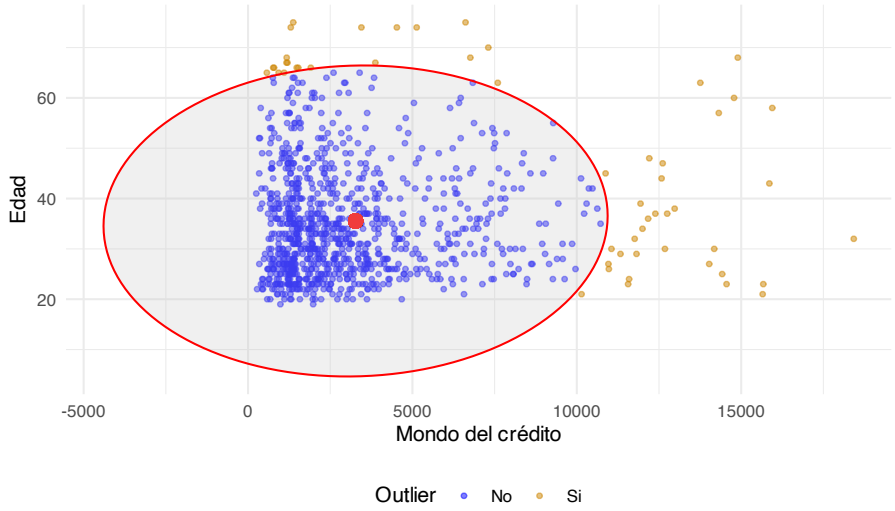
Ahora podemos incluir una elipse con el umbral para la **distancia de Mahalanobis** para visualizar los *outliers* como en la Figura 4.6. El código se ha omitido intencionalmente.

Este tipo de visualizaciones como la Figura 4.6 es muy común cuando solo se cuenta con dos variables, pero en el *business analytics* es muy rara la situación en la que solo contemos con dos *features*. Si deseas obtener una visualización como la Figura 4.6 sin tener que construirla manualmente (como fue nuestro caso), podemos emplear la función **drawMahal()** que viene en el paquete *chemometrics* (Filzmoser, 2023).

La función **drawMahal()** emplea cuatro argumentos principalmente:

drawMahal(x, center, covariance, quantile)

Figura 4.6. Edad y monto de crédito para los clientes del banco con umbral de la distancia de Mahalanobis.



Fuente: elaboración propia.

donde:

- **x**: los datos multivariados que pueden ser de clase **matrix** o **data.frame**.
- **center**: El centro de los datos; es decir, los valores medios de las variables.
- **covariance**: La matriz de varianzas y covarianzas de los datos.
- **quantile**. Los cuantiles de distancia a graficar. Recuerda que Cabana et al. (2021) sugieren emplear α en 0,025; es decir, emplear el umbral en el 2,5 % de observaciones más extremas. que equivale a emplear el cuantil 0.975 (probabilidad de 97.5 %) Es decir, cualquier observación fuera de 97.5 % de probabilidad se considerará un valor atípico.

En este caso, el código para tener todos los elementos necesarios para crear la Figura 4.7 es:

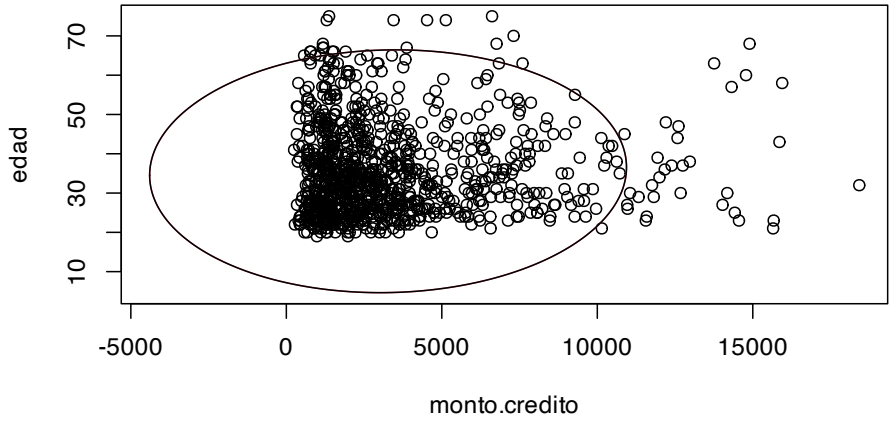
```
# install.packages('chemometrics')
library(chemometrics)
# Seleccionar las variables a analizar
datos_mah <- german_cuanti[c("monto.credito", "edad")]

# Encontrar el punto central
datos_mah.center = colMeans(datos_mah)

# Encontrar la matriz de covarianzas
datos_mah.cov = cov(datos_mah)
```

Ahora sí podemos emplear la función **drawMahal()** para obtener la Figura 4.7. El

código correspondiente es:

```
drawMahal(datos_mah, center = datos_mah.center, covariance = datos_mah.cov,
↪  quantile = 0.975)
```

Figura 4.7. Edad y monto de crédito para los clientes del banco con umbral de la distancia de Mahalanobis creada automáticamente.



Fuente: elaboración propia.

Nota que en este caso no hemos calculado la **distancia de Mahalanobis** con estimadores robustos. Podríamos emplear la función **covMcd()** del paquete *robustbase* (Maechler et al., 2024). para estimar la media y la varianza robustas empleando el método *MCD* y reemplazar estos valores en los argumentos **center** y **covariance** de la función **drawMahal()**. Para calcular la matriz de varianzas y covarianzas robusta por el método *EVM*, podemos emplear la función **cov.rob()** del paquete *MASS* (Venables y Ripley, 2002), con el argumento **method = "mve"**. ¡Inténtalo!

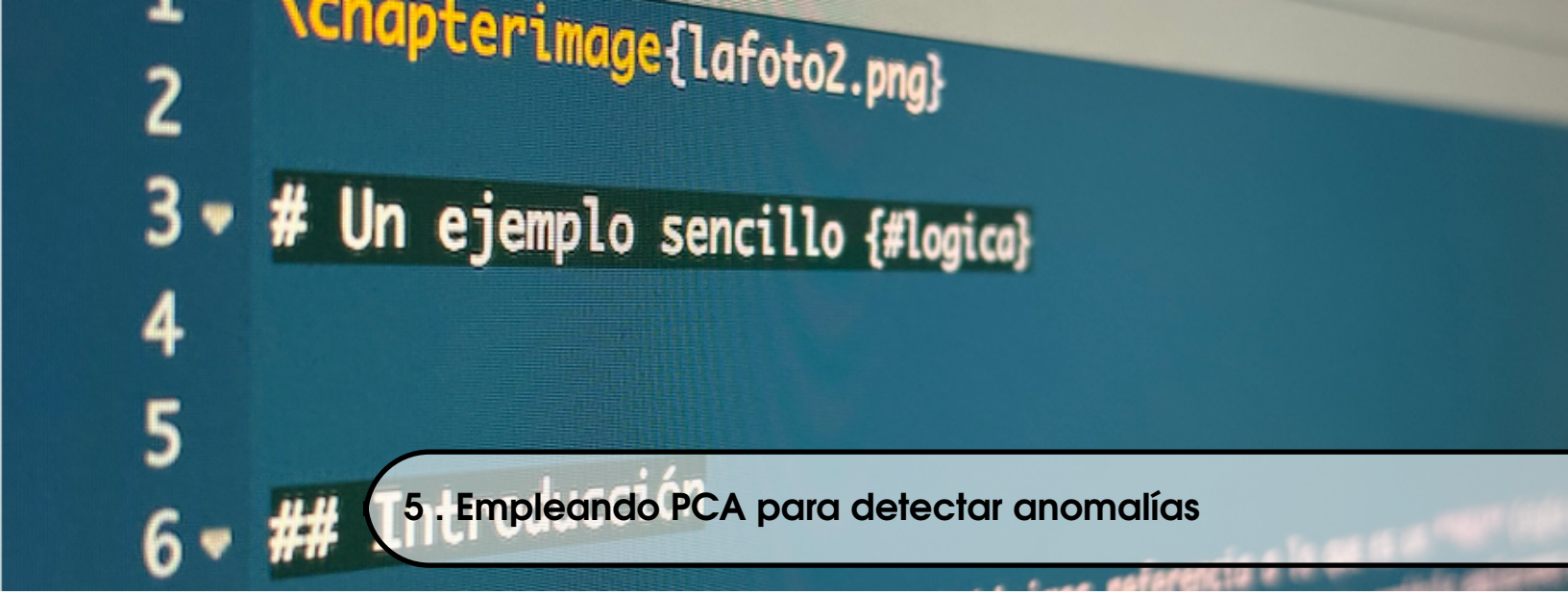
4.5 Comentarios finales

En este capítulo estudiamos cómo emplear la **distancia de Mahalanobis** para detectar anomalías multivariadas. La distancia de Mahalanobis y sus variantes robustas permiten identificar *outliers* multivariados. Su principal fortaleza es capturar relaciones entre variables, pero su aplicación depende de que los datos tengan una estructura aproximadamente elíptica. Por ello, conviene usarlas en conjunto con técnicas exploratorias vistas en capítulos previos. Estas técnicas, al igual que las estudiadas en los Capítulos 2 y 3, permiten encontrar anomalías que se pueden caracterizar como anomalías globales (puntuales); es decir, *outliers*.

Es importante recalcar que tanto las aproximaciones multivariadas como las univariadas no son mutuamente excluyentes. En el proceso de **ETL** es muy común el uso de técnicas univariadas para la detección de *outliers*, por otro lado, es en el proceso de **EDA** en el que el análisis de outliers multivariados como los estudiados en este capítulo

y en 5 es más común. Las herramientas estudiadas hasta aquí se deben entender como complementarias y parte de la caja de herramientas que todo científico de datos debe tener a la mano, independientemente del tipo de preguntas de negocio que estemos intentando responder.

En este capítulo ampliamos la caja de herramientas para la detección de valores atípicos. En los Capítulos 5 y 6 emplearemos más herramientas con un origen en la estadística. Para posteriormente adentrarnos en las técnicas de detección de anomalías con un origen en el aprendizaje de máquina.



5 . Empleando PCA para detectar anomalías

5.1 Introducción

En los Capítulos 2 y 3 estudiamos métodos de origen en la estadística para detectar *outliers* (anomalías univariadas puntuales). Adicionalmente, en el Capítulo 4 discutimos cómo encontrar *outliers multivariados* . En este capítulo estudiaremos una técnica para detectar anomalías multivariadas empleando el análisis de **Componentes Principales** (**PCA** por las siglas del término en inglés *Principal Component Analysis*)[1] El PCA es una técnica **matemática** que permite resumir la información contenida en muchas variables en unas pocas. Si bien esta técnica es de origen en las matemáticas, la estadística la ha adoptado como propia y hoy hace parte de las herramientas convencionales de todo usuario de la estadística que emplea muchas variables (muchas dimensiones en la jerga del **PCA**). A este procedimiento se le conoce técnicamente como **reducción de la dimensionalidad de los datos.**

El **PCA** es un procedimiento matemático que no requiere supuestos, ni siquiera sobre la distribución probabilística de los datos, que busca reducir las dimensiones. En este capítulo primero estudiaremos la intuición detrás del procedimiento (Sección 5.2), para después concentrarnos en el detalle técnico del **PCA** (Sección 5.3)) para posteriormente mostrar en la Sección 5.4 cómo se emplea esta técnica para encontrar *outliers*. Posteriormente, estudiaremos cómo aplicar este procedimiento en R (Sección 5.5.1).

5.2 La intuición detrás del PCA

Por un momento, imaginémonos que contamos con una base de datos (`data.frame`) que tiene d variables y para cada variable tenemos n observaciones.

[1] Partes de este documento provienen de unas notas de clase que se publicaron en Alonso (2020). En dicho documento se presenta con mayor detalle cómo funciona la técnica del PCA.

$$\begin{matrix} x_{1,1} & x_{2,1} & \cdots & x_{d,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{d,2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1,n} & x_{2,n} & \cdots & x_{d,n} \end{matrix} \tag{5.1}$$

En este caso, estamos empleando el primer subíndice para denotar el número de la variable y el segundo el número de la observación. Esta notación sigue el estándar de la estadística para denotar matrices de datos, donde las columnas son las variables y las filas son las observaciones. Esta es la misma notación que se emplea típicamente en el contexto de la regresión múltiple (Alonso, 2024). En otras palabras, $x_{j,i}$ representa la observación i-ésima de la variable j, donde $j = 1, 2, \cdots d$ (tenemos d variables) y $i = 1, 2, \cdots n$ (tenemos n observaciones). No sigue el estándar del álgebra matricial donde el primer subíndice corresponde a la fila y el segundo a la columna.

Nota que al calcular estadísticas descriptivas lo que hacemos es concentrarnos en una columna de la base de datos y resumirla con un número. Las entradas de la base de datos corresponden a observaciones que se emplean para estimar (encontrar) el estadístico descriptivo (como la media[2]) que no es observable.

El **PCA** es algo diferente, a partir de varias columnas (variables) observadas se quiere encontrar otras variables no observables que las resuman. Esas variables no observables se conocen como componentes principales. La Figura 5.1 presenta un diagrama de esta idea.

En otras palabras, tras el **PCA** no tendremos las d variables originales, sino que tendremos p componentes principales que resumen las d variables originales. Es decir, ahora tendremos:

$$\begin{matrix} y_{1,1} & y_{2,1} & \cdots & y_{p,1} \\ y_{1,2} & y_{2,2} & \cdots & y_{p,2} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,n} & y_{2,n} & \cdots & y_{p,n} \end{matrix} \tag{5.2}$$

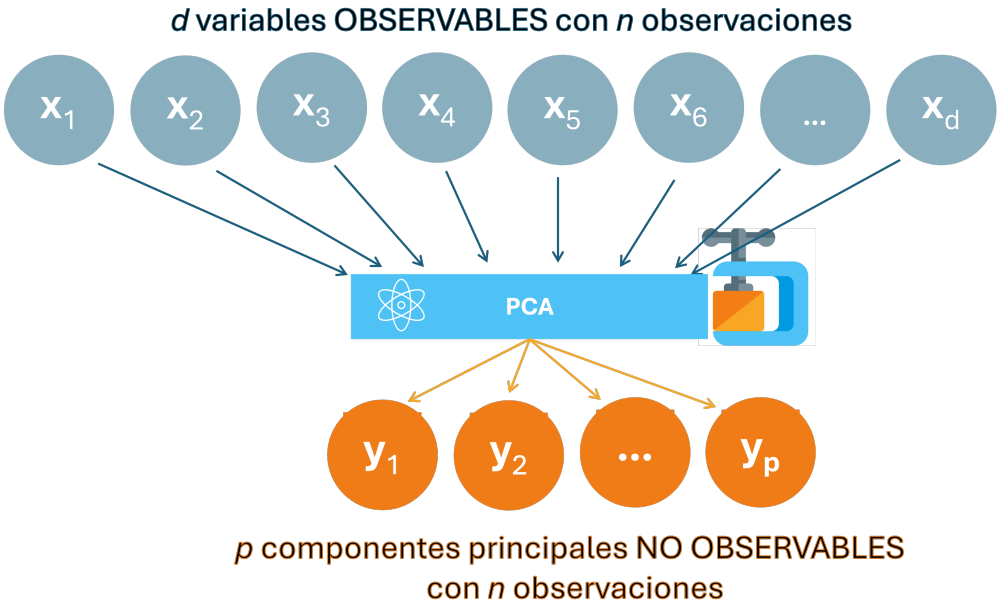
Para que el ejercicio de **PCA** tenga sentido, se espera que p sea sustancialmente menor que d.

5.3 Detalle técnico del PCA

Formalmente, queremos partir de una matriz de datos con las d variables observadas que se desean resumir. Es decir,

[2]Por ejemplo, la media de la variable j sería $\bar{x_j} = \frac{\sum_{i=1}^{n} x_{j,i}}{n}$.

Figura 5.1. Representación del PCA



Fuente: adaptación de Alonso (2020).

Figura 5.2. Material multimedia: PCA

Fuente: elaboración propia. https://ln.run/MHDAW

$$\mathbf{X}_{n\times d} = \begin{bmatrix} x_{1,1} & x_{2,1} & \cdots & x_{d,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{d,2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1,n} & x_{2,n} & \cdots & x_{d,n} \end{bmatrix} \tag{5.3}$$

Y la tarea será encontrar p componentes principales. En otras palabras debemos encontrar

$$\mathbf{Y}_{n\times p} = \begin{bmatrix} y_{1,1} & y_{2,1} & \cdots & y_{p,1} \\ y_{1,2} & y_{2,2} & \cdots & y_{p,2} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,n} & y_{2,n} & \cdots & y_{p,n} \end{bmatrix} \tag{5.4}$$

Esto implica encontrar una matriz $\mathbf{W}_{d\times p}$ que cumpla la siguiente condición:

$$\mathbf{Y}_{n\times p} = \mathbf{X}_{n\times d}\mathbf{W}_{d\times p} \tag{5.5}$$

La matriz $\mathbf{W}_{d\times p}$ es conocida como la matriz de cargas (*loadings*). Donde,

$$\mathbf{W}_{d\times p} = \begin{bmatrix} \gamma_{1,1} & \gamma_{2,1} & \cdots & \gamma_{p,1} \\ \gamma_{1,2} & \gamma_{2,2} & \cdots & \gamma_{p,2} \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_{1,d} & \gamma_{2,d} & \cdots & \gamma_{p,d} \end{bmatrix} \tag{5.6}$$

De esta manera, cada uno de los p componentes principales para la i-ésima observación será:

$$\begin{aligned} y_{1,i} &= \gamma_{1,1}x_{1,i} + \gamma_{1,2}x_{2,i} + \gamma_{1,3}x_{3,i} + \vdots + \gamma_{1,p}x_{d,i} \\ y_{2,i} &= \gamma_{2,1}x_{1,i} + \gamma_{2,2}x_{2,i} + \gamma_{2,3}x_{3,i} + \vdots + \gamma_{2,p}x_{d,i} \\ &\vdots \\ y_{p,i} &= \gamma_{p,1}x_{1,i} + \gamma_{p,2}x_{2,i} + \gamma_{p,3}x_{3,i} + \vdots + \gamma_{p,d}x_{d,i} \end{aligned} \tag{5.7}$$

Las cargas $\gamma_{j,l}$ se interpretan como el aporte de la variable l al componente principal j. Nota que los p componentes principales son una combinación lineal de las variables y los γs son las ponderaciones de cada variable en el respectivo componente principal.

Ahora, la pregunta será cómo encontramos la matriz $\mathbf{W}_{d\times p}$. La respuesta es relativamente sencilla si se emplean resultados del álgebra matricial. Para encontrar los componentes principales se debería buscar aquellos que puedan explicar la mayor cantidad de la varianza de las variables originales. Y dado que en este caso se trata de más de una variable, pues en últimas lo que queremos es poder explicar la mayor parte de la matriz de varianzas y covarianzas de los datos originales ($\Sigma_{d\times d}$), esta aproximación se conoce como la maximización de la varianza. Por otro lado, es muy fácil demostrar que maximizar la varianza explicada es equivalente a encontrar los factores propios que impliquen la menor distancia promedio elevada al cuadrado (media

cuadrática) entre los componentes principales y los vectores (variables) originales. A este método se le conoce como minimizar el error cuadrado. En cualquiera de los dos casos, para p factores propios se tiene que la matriz $\mathbf{W}_{d\times p}$ implica encontrar los p vectores y valores propios de la matriz de varianzas y covarianzas de los datos originales ($\Sigma_{d\times d}$). Esto permitirá encontrar p factores propios que serán ortogonales entre sí. Es decir, cada una de las nuevas variables creadas será independiente entre sí.

Antes de continuar, es importante mencionar que no es común que todas las variables originales se midan en las mismas unidades. Por eso, una buena práctica es estandarizar las variables para dar la misma influencia a cada una de ellas.

Para encontrar los p valores propios que resuman las d variables originales se tendrán que seguir los siguientes pasos:

1. Estandarizar los datos originales. Por simplicidad, llamemos a estos datos $\mathbf{X}_{n\times d}$.
2. Calcular la matriz de varianzas y covarianzas de los datos originales. En otras palabras, a partir de $\mathbf{X}_{n\times d}$, encontrar la matriz de varianzas y covarianzas $\Sigma_{d\times d}$.
3. Calcular todos los vectores propios y sus valores correspondientes.
4. Ordenar los vectores propios de acuerdo con sus valores propios en orden decreciente.
5. Elegir los primeros p vectores propios y esas serán las nuevas p dimensiones.
6. Transformar las n observaciones de $\mathbf{X}_{n\times d}$ en los nuevos p componentes principales $\mathbf{Y}_{n\times p}$.

El **PCA** solo se puede realizar con variables cuantitativas. Una de las tareas importantes al realizar un **PCA** es determinar el número de componentes principales (p) (paso 5). Para esto típicamente se emplean uno de los siguientes tres criterios (Ver Alonso (2020) para una mayor discusión) para determinar el "mejor" número de componentes (en inglés se conocen como *stopping rules*):

- criterio del codo (o también conocido como *scree test*),
- la regla de Keisser-Guttman y
- el análisis paralelo (también conocido como análisis paralelo de Horn).

El criterio del codo implica crear un gráfico de codo (*elbow graph*, también conocido como "scree plot") que muestra la proporción de la varianza de todos los datos explicada por cada uno de los componentes principales. Esto se hace graficando en el eje horizontal el número de componentes principales y en el vertical la división de la varianza de cada componente principal entre la varianza total de todos los componentes principales.

El gráfico de codo muestra cómo disminuye la varianza explicada a medida que aumenta el número de componentes. En ese gráfico debemos identificar el "codo", entendido como el punto donde la disminución de la varianza explicada se hace menos pronunciada; es decir, donde la curva comienza a aplanarse. Este punto sugiere que añadir más componentes principales no proporciona una mejora significativa en la explicación de la variabilidad de los datos.

De esta manera, el número de componentes principales donde se presenta el "codo" será considerado como el número óptimo de componentes (p).

El método del codo es considerado como una técnica heurística y "subjetiva" pues depende de la experiencia del científico de datos para detectar el "codo". Es más, en algunos casos puede no haber un "codo" evidente.

Por otro lado, la regla de Keisser-Guttman implica mantener aquellos componentes cuyos valores propios son superiores a uno. La idea detrás de esto es que los valores propios representan la proporción de la varianza total descrita por cada componente. Así, al mantener solo aquellos mayores a uno, estamos reteniendo los componentes que explican más de la variabilidad que las variables originales.

Otra forma de determinar el número óptimo de componentes principales fue sugerida por Horn (1965). El análisis paralelo, también conocido como análisis paralelo de Horn, es un método estadístico que nos permite determinar p. El método compara los valores propios generados a partir de los datos originales con los valores propios generados a partir de una matriz simulada (se realizan simulaciones de Monte-Carlo) creada a partir de datos aleatorios del mismo tamaño (para mayor detalle consultar a Horn (1965)). Al igual que el criterio anterior se retienen aquellos componentes con valores propios simulados (ajustados) mayores a uno.

5.4 PCA para detectar anomalías

Hasta el momento hemos estudiado cómo realizar un **PCA**, pero no hemos discutido cómo emplear el **PCA** para detectar anomalías en una muestra. Una vez tenemos las n observaciones de las d variables ($\mathbf{X}_{n\times d}$) transformadas en los nuevos p componentes principales ($\mathbf{Y}_{n\times p}$), una de las prácticas más comunes es visualizar los valores de los componentes principales (típicamente los dos primeros componentes principales) con un gráfico de dispersión, teniendo en el eje horizontal el primer componente y en el vertical el segundo componente. Los puntos que están muy alejados de la mayoría de los puntos pueden considerarse *outliers*. Otra aproximación gráfica es visualizar con *boxplots* los valores de cada uno de los componentes principales para cada observación.

Otros autores sugieren no graficar directamente los componentes principales, sino la **distancia Mahalanobis**. Es decir, después de realizar el **PCA**, se calcula la distancia Mahalanobis de cada observación al centroide (o centro) de los datos en el espacio de las componentes principales (En la Sección 4.3.1 discutimos esta distancia en detalle). Las observaciones que tienen una distancia Mahalanobis alta pueden considerarse *outliers*.

Como se discutió en la Sección 4.3.1, la distancia de Mahalanobis, se define como:

$$d(\mathbf{y_i}) = \sqrt{(\mathbf{y_i} - \bar{\mathbf{Y}})^T \Sigma^{-1} (\mathbf{y_i} - \bar{\mathbf{Y}})} \tag{5.8}$$

en este caso, $\mathbf{y_i}$ representa el vector de los valores que toman los componentes principales para la observación i-ésima; en otras palabras, la i-ésima fila de la matriz $\mathbf{Y}_{n\times p}$. Por otro lado, $\bar{\mathbf{Y}}$ corresponde al vector que contiene las medias para cada compo-

nente principal; es decir, la media de cada columna de la matriz $\mathbf{Y}_{n\times p}$. Y finalmente,

$$\Sigma = \begin{bmatrix} S_1^2 & S_{1,2} & \cdots & S_{1,d} \\ S_{1,2} & S_2^2 & \cdots & S_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ S_{1,d} & S_{2,d} & \cdots & S_d^2 \end{bmatrix} \tag{5.9}$$

donde $S_{k,l}$ es la covarianza entre el k-ésimo componente principal (columna k de la matriz $\mathbf{Y}_{n\times p}$) y el l-ésima componente principal y S_k^2 es la varianza del k-ésima componente principal.

Es importante recordar (como lo estudiamos en la Sección 4.3.1) que la distancia euclidiana solo toma en cuenta la magnitud de las diferencias entre las coordenadas de dos puntos, sin considerar la correlación entre las variables. En cambio, la distancia de Mahalanobis incorpora la información de la matriz de covarianza, ponderando las diferencias entre las variables de acuerdo con su importancia y correlación. Adicionalmente, se ha documentado que la distancia de Mahalanobis es menos sensible a la presencia de *outliers* o valores extremos en el conjunto de datos. Esto la convierte en una medida de distancia muy empleada en muestras multivariables (multidimensionales) donde las variables están interrelacionadas y existen *outliers*.

Otra forma de emplear el análisis de **PCA** para la detección de anomalías es construir un **Score de Anomalía**. Esta aproximación implica calcular una puntuación de anomalía para cada observación de la siguiente manera:

$$Score_i = \sum_{j=1}^{p} \left(\frac{y_{j,i}^2}{\lambda_j} \right) \tag{5.10}$$

con λ_j representando el valor propio (**eigenvalue**) asociado con el j-ésimo componente principal. Recuerda que $Y_{j,i}$ es el valor del j-ésimo componente principal para la observación i (originalmente x_i).

Las observaciones con *scores* más altos se consideran que tienen una probabilidad más alta de ser *outliers*. El uso de los *scores* de anomalía en lugar de medidas directas de distancia como la distancia Mahalanobis de los componentes principales puede ser útil porque tiene en cuenta la varianza explicada por cada componente principal, lo que puede ayudar a identificar *outliers* en conjuntos de datos multidimensionales de manera más efectiva. El problema que trae emplear estos *scores* de anomalía es que no existe un umbral claro para establecer que una observación es un *outlier*.

5.5 Implementación en R

Antes de realizar la detección de anomalías empleando el **PCA**, veamos en detalle cómo realizar un **PCA** en R. Si ya dominas esta técnica, puedes saltarte esta subsección. Si quieres estudiar en más detalle el **PCA**, puedes consultar Alonso (2020).

5.5.1 PCA en R

Para realizar un ejemplo práctico, seguiremos usando los mismos datos que empleamos en los Capítulos 2, 3 y 4. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalias). La descripción de las variables se encuentra en la Sección 2.4. Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Seleccionemos solo las variables cuantitativas para poder aplicar las técnicas estudiadas en las secciones anteriores. Empleemos la función **select_if()** del paquete *dplyr* (Wickham et al., 2021) para hacer esta tarea más sencilla.

```
library(dplyr)
german_cuanti <- german %>%
    select_if(is.numeric)
```

Es decir, en este caso contamos con una matriz $\mathbf{X}_{n\times d}$ de dimensiones 1000×6 (mil observaciones y 6 variables).

En R es posible realizar el PCA con diferentes paquetes como *psych* (William Revelle, 2023), *ade4* (Bougeard y Dray, 2018) , *amap* (Lucas, 2022) , *ca* (Nenadic y Greenacre, 2007) y *MASS* (Venables y Ripley, 2002) , aún con la función **princomp()** del core de R. Nosotros emplearemos el paquete *FactoMineR* (Lê et al., 2008) .

La función **PCA()** del paquete *FactoMineR* típicamente incluye los siguientes argumentos:

PCA(X, scale.unit, ncp, quanti.sup, quali.sup, graph, axes

donde:

- **X**: La matriz de datos originales $\mathbf{X}_{n\times d}$ (sin estandarizarse).
- **scale.unit**: Si **scale.unit = TRUE** (valor por defecto), los datos son estandarizados.
- **ncp**: p. Número de factores propios.
- **graph**: Si **graph = TRUE** (opción por defecto), se presenta un gráfico (biplot) de dispersión para cada variable incluida en el análisis y con dos componentes principales en cada uno de los ejes.
- **axes**: Un vector de tamaño dos que especifica qué componentes principales serán graficados.

Empleemos esta función para calcular inicialmente el PCA con $p = 6$. Naturalmente, esto no ayuda a reducir las dimensiones, pero si nos permitirá escoger en un paso posterior el mejor p.

```
library(FactoMineR)
pca_inicial <- PCA(german_cuanti, graph = FALSE)
summary(pca_inicial)
```

```
##
## Call:
## PCA(X = german_cuanti, graph = FALSE)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6
## Variance               1.414   1.285   0.993   0.886   0.728   0.695
## % of var.             23.567  21.423  16.543  14.762  12.129  11.575
## Cumulative % of var.  23.567  44.991  61.533  76.296  88.425 100.000
##
## Individuals (the 10 first)
##                          Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2
## 1                     |  3.374 |  2.585  0.472  0.587 | -1.468  0.168  0.189 |
## 2                     |  2.085 | -1.587  0.178  0.579 |  1.300  0.132  0.389 |
## 3                     |  2.881 |  1.211  0.104  0.177 |  0.682  0.036  0.056 |
## 4                     |  3.341 |  1.654  0.194  0.245 |  2.010  0.314  0.362 |
## 5                     |  3.206 |  2.819  0.562  0.773 |  0.634  0.031  0.039 |
## 6                     |  3.465 |  1.143  0.092  0.109 |  2.341  0.426  0.457 |
## 7                     |  2.039 |  1.053  0.078  0.267 | -0.381  0.011  0.035 |
## 8                     |  1.930 | -0.848  0.051  0.193 |  1.470  0.168  0.580 |
## 9                     |  2.748 |  1.441  0.147  0.275 |  0.265  0.005  0.009 |
## 10                    |  1.896 | -0.409  0.012  0.046 | -0.153  0.002  0.007 |
##                        Dim.3    ctr   cos2
## 1                     -0.751  0.057  0.050 |
## 2                     -0.227  0.005  0.012 |
## 3                      1.174  0.139  0.166 |
## 4                      0.220  0.005  0.004 |
## 5                      1.058  0.113  0.109 |
## 6                      0.277  0.008  0.006 |
## 7                     -1.349  0.183  0.437 |
## 8                     -0.554  0.031  0.082 |
## 9                     -1.502  0.227  0.299 |
## 10                     0.406  0.017  0.046 |
##
## Variables
##                          Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## monto.credito         |  0.098  0.681  0.010 |  0.762 45.171  0.581 | -0.282
## porcentaje.disponible |  0.074  0.384  0.005 | -0.805 50.414  0.648 | -0.009
## residente             |  0.645 29.451  0.416 | -0.089  0.621  0.008 | -0.486
## edad                  |  0.738 38.494  0.544 | -0.067  0.350  0.005 | -0.198
## numero.creditos       |  0.522 19.285  0.273 |  0.012  0.011  0.000 |  0.393
## dependientes          |  0.407 11.703  0.165 |  0.210  3.433  0.044 |  0.695
##                          ctr   cos2
## monto.credito          8.015  0.080 |
## porcentaje.disponible  0.009  0.000 |
```

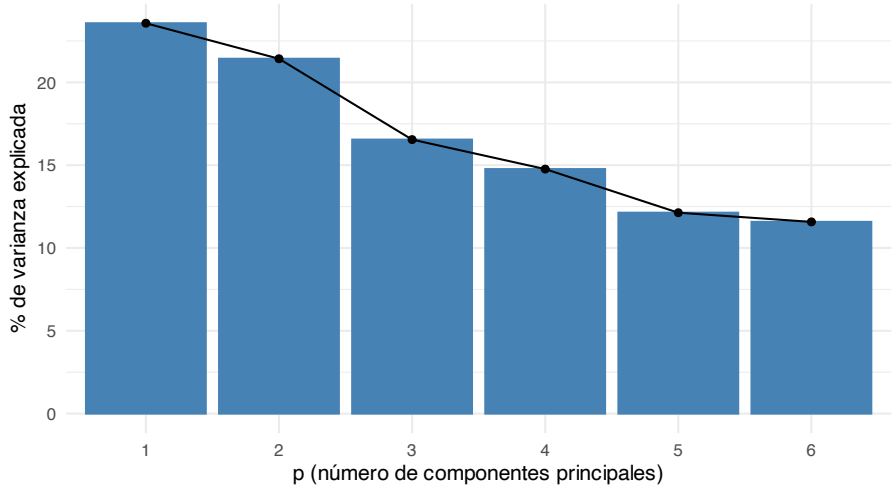
```
## residente              23.826  0.236 |
## edad                    3.931  0.039 |
## numero.creditos        15.557  0.154 |
## dependientes           48.662  0.483 |
```

En el compartimiento `$eig` podemos ver los valores propios (*eigenvalue*) que están organizados de mayor a menor, el porcentaje de varianza explicado por cada componente (*percentage of variance*) y el porcentaje acumulado de varianza (*cumulative percentage of variance*). Por ejemplo, el primer componente explica el 23.57 % de la varianza y los primeros tres componentes explican el 61.53 % de la varianza.

Ahora decidamos el número de componentes óptimo (p). El gráfico de codo o *scree plot* se puede construir fácilmente empleando la función **fviz_screeplot()** del paquete *factoextra* (Kassambara y Mundt, 2020). .

```
library(factoextra)
fviz_screeplot(pca_inicial) + labs(title = "", x = "p (número de componentes
↪  principales)",
    y = "% de varianza explicada") + theme_minimal()
```

Figura 5.3. Gráfico de codo



Fuente: elaboración propia.

Recuerda que nuestro objetivo es encontrar los componentes que explican la mayor varianza. Pues, queremos conservar tanta información como sea posible utilizando el menor número de componentes. Entonces, cuanto mayor sea la varianza explicada, mayor será la información contenida en esos componentes.

De la Figura 5.3 podemos ver que el primer componente principal explica el 23.57 % de la varianza. El segundo componente explica el 21.42 %. Y así sucesivamente. En la Figura 5.3 debemos buscar el **codo** (elbow): el punto donde la curva se aplana. Es decir, donde no se empieza a agregar mucho a la varianza. En este caso, el **codo** aparece

en el quinto componente y, por tanto, esto implicaría usar solo los componentes a la izquierda; es decir, 4 componentes. Con esos cuatro componentes podemos explicar el 76.3 % de la varianza. Pero esto significaría una reducción muy baja de la dimensionalidad; de 6 a 4 variables.

Por otro lado, la regla de Keisser-Guttman implica mantener aquellos mayores a uno. Recordemos que los valores propios los podemos ver de manera muy sencilla con el siguiente código:

```
pca_inicial$eig
```

```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1  1.4140252               23.56709                          23.56709
## comp 2  1.2854088               21.42348                          44.99057
## comp 3  0.9925758               16.54293                          61.53350
## comp 4  0.8857343               14.76224                          76.29573
## comp 5  0.7277376               12.12896                          88.42469
## comp 6  0.6945183               11.57531                         100.00000
```

Esta regla implicaría emplear 2 componentes. Ahora sí, sería una reducción sustancial de la nacionalidad.

Por otro lado, el análisis paralelo o análisis paralelo de Horn se puede implementar en R por medio de la función **paran()** del paquete *paran* (Dinno, 2018). Esta función necesita como argumentos los datos originales (la estandarización se hace de manera automática), si se desea una semilla para los cálculos aleatorios se puede fijar con el argumento **seed** y un argumento lógico para mostrar el gráfico (**graph**).

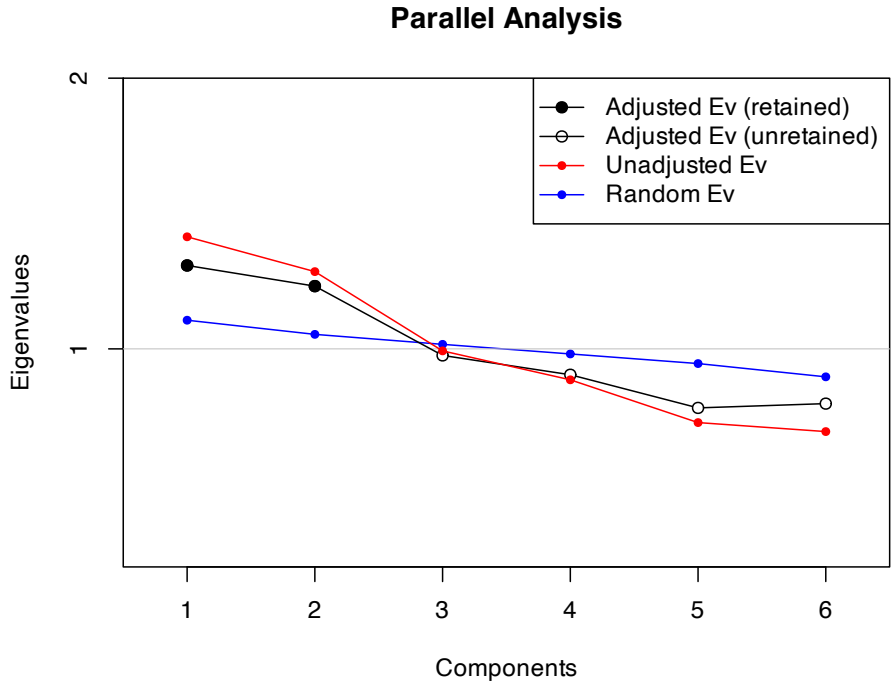
En este caso, el código con el que guardamos los resultados del análisis paralelo en el objeto `pca_paralelo` y creamos la Figura 5.4 es el siguiente:

```
library(paran)
pca_paralelo <- paran(german_cuanti, graph = TRUE, seed = 0)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## ----------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ----------------------------------------------------
## 1           1.308144    1.414025      0.105880
## 2           1.231785    1.285408      0.053623
## ----------------------------------------------------
```

```
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

Figura 5.4. Análisis paralelo de Horn para encontrar el número óptimo de componentes principales



Fuente: elaboración propia.

En la Figura 5.4, los componentes principales "retenidos" (óptimos) se indican con marcadores circulares sólidos en la línea de color negro que corresponde al valor propio ajustado (*Adjusted Ev*) y los componentes principales no "retenidos" se indican con marcadores circulares sin relleno.

El número óptimo de componentes principales lo podemos encontrar en el compartimiento `$Retained` del objeto creado. Es decir,

```
pca_paralelo$Retained
```

```
## [1] 2
```

Este criterio sugiere mantener los 2 primeros componentes. De esta manera, tenemos que el criterio del codo sugiere 4 componentes, mientras que la regla de Keisser-Guttman y el análisis paralelo sugieren 2 componentes. Trabajaremos con 2 componentes que corresponden a una reducción sustancial de la dimensionalidad. Es importante mencionar que el análisis paralelo es considerado como una aproximación más sofisticada que las otras dos.

Ya hemos determinado que $p = 2$. Calculemos el **PCA** para dos componentes.

```
pca_final <- PCA(german_cuanti, graph = FALSE, ncp = 2)
```

Finalmente, el valor de los componentes principales lo podemos encontrar en el compartimiento `$ind[1]` del objeto creado.

5.5.2 Detencción de anomalías con el PCA en R

Ya hemos encontrado los componentes principales; es decir, hemos podido expresar las observaciones que tenemos para los mil individuos de las seis variables cuantitativas en dos componentes principales para cada uno de los individuos. Ahora la tarea es emplear esos componentes principales para detectar las observaciones anormales.

Típicamente, el primer paso es visualizar en un diagrama de dispersión los componentes principales encontrados para cada observación. Esto lo podemos hacer de muchas maneras. Una forma muy fácil es emplear la función **fviz_pca_ind()** del paquete *factoextra* (Kassambara y Mundt, 2020). Esta función solo necesita como argumentos un objeto de clase **PCA** (objeto donde guardamos los resultados del **PCA**) y determinar cuáles componentes principales se desean graficar. En nuestro caso solo encontramos dos componentes principales, pero en otras ocasiones podremos tener más y será interesante no solo graficar los primeros dos componentes principales sino por ejemplo el primero y el segundo. En nuestro caso, que solo queremos ver los dos primeros componentes principales, emplearemos **axes = c(1, 2)**. El código sería el siguiente (intencionalmente no se muestra el gráfico resultante):

```
fviz_pca_ind(pca_final, axes = c(1, 2))
```

Es también común incluir en esta visualización una elipse llamada **elipse de confianza** o **elipse de dispersión**. La elipse generalmente representa un intervalo de confianza alrededor de los datos (comúnmente del 97,5 %) teniendo en cuenta la distancia de Mahalanobis (Para una discusión sobre este tema, ver la Sección 4.4). Los puntos que caen fuera de la elipse pueden considerarse *outliers* en el espacio de las componentes principales.

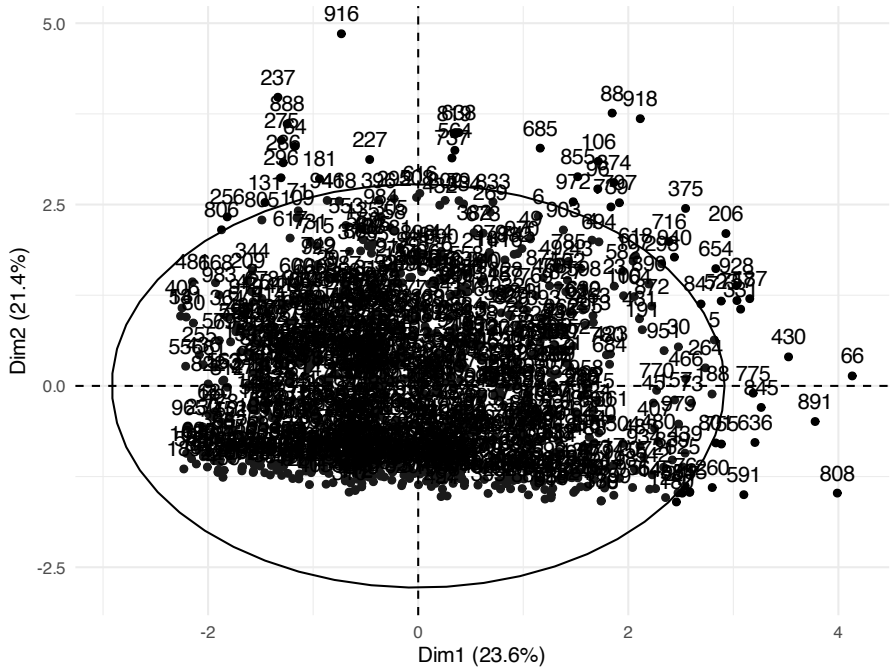
La **elipse de dispersión** la podemos construir con el argumento **addEllipses = TRUE**. El código para generar la Figura 5.4 es el siguiente:

```
fviz_pca_ind(pca_final, axes = c(1, 2), addEllipses = TRUE)
```

En la Figura 5.4 cada punto representa una observación (i) y el número corresponde al número de la observación. Se puede observar que el individuo 916 tiene un valor anormalmente grande de la dimensión 2 (segundo componente principal) y el 66 en la dimensión 1.

Por otro lado, si queremos emplear la **distancia de Mahalanobis**, como se discutió en la Sección 5.4, existe la función **pca.outlier()** del paquete *mt* (Lin, 2024). Esta función detecta automáticamente los valores atípicos empleando las distancias de Mahalanobis de los primeros dos componentes principales. Adicionalmente, permite

Figura 5.5. Diagrama de dispersión para los dos componentes principales y su respectiva elipse de dispersión



Fuente: elaboración propia.

visualizar los componentes principales 1 y 2 en un gráfico de dispersión con su elipse de confianza, empleando la distancia de Mahalanobis.

La función típicamente incluye los siguientes argumentos:

pca.outlier(x, center, scale, conf.level)

donde:

- **x**: Es un objeto de clase **matrix** o **data.frame** que contiene los datos originales $\mathbf{X}_{n\times d}$ (sin estandarizarse).
- **center**: Si **center = TRUE** (valor por defecto) se le resta a los datos la media (se centran) para que tengan una media cero antes del **PCA**.
- **scale**: Si **scale = TRUE** (valor por defecto) se escalan los datos para que cada variable tenga varianza 1 antes del **PCA**.
- **conf.level**: El nivel de confianza para controlar el corte de las distancias de Mahalanobis. Es decir, para la construcción de las elípses.

```
# Cargar paquete
library(mt)

outlier_mahala <- pca.outlier(german_cuanti, center = TRUE, scale = TRUE,
↪  conf.level = 0.975)
```

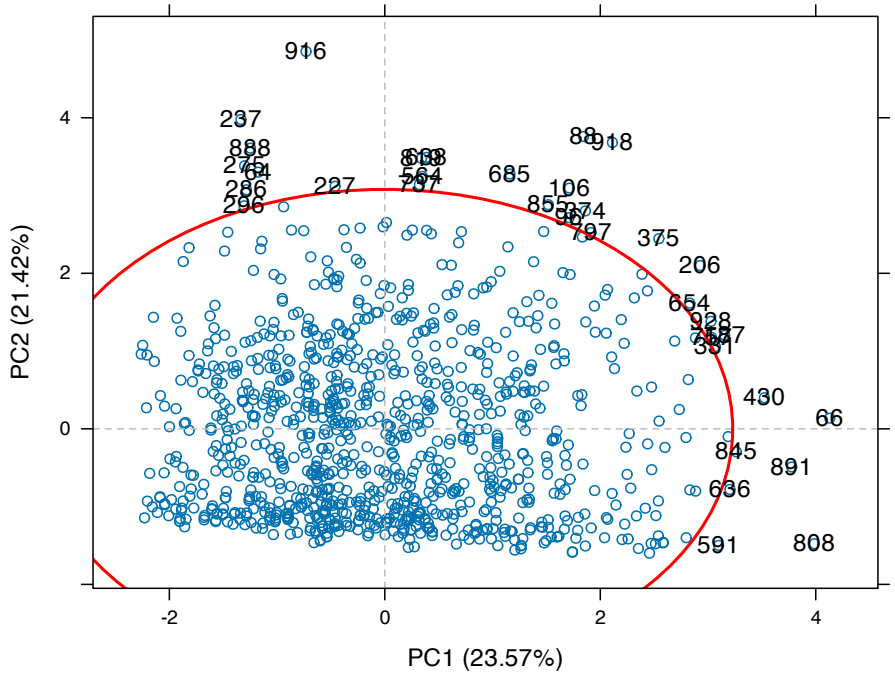
En el objeto `outlier_mahala` podemos encontrar en el compartimiento `$outlier` los puntos que quedan por fuera de la elipse de dispersión; es decir, los *outliers*. Y si invocas el objeto `outlier_mahala` podrás ver la visualización de los componentes principales 1 y 2 en un gráfico de dispersión con su elipse de confianza, empleando la distancia de Mahalanobis, tal como se presenta en la Figura 5.6.

De acuerdo con esta aproximación, tenemos 34 *outliers*.

Finalmente, para calcular el *score* de anomalías, según la expresión (5.10), hemos construido la siguiente función:

```
score.anomal.PCA <- function(X, p) {
    # X = objeto de clase PCA con el análisis de PCA p = número de Componentes
    # principales

    eig <- t(as.matrix(X$eig[1:p, 1]))  # Extraer los valores propios
    CP <- X$ind[1]  # Extraer los componentes principales
    CP <- as.matrix(CP$coord)
    temp <- CP^2/matrix(data = rep(eig, dim(CP)[1]), ncol = p)
    scores <- apply(CP^2/matrix(data = rep(eig, dim(CP)[1]), ncol = p), 1, sum)
    scores <- as.data.frame(scores)

    scores <- cbind(rownames(scores), scores)
    names(scores) <- c("Id de la observación", "Score de anomalía")

    scores <- scores[order(scores$`Score de anomalía`, decreasing = TRUE), ]
```

Figura 5.6. Diagrama de dispersión para los dos componentes principales y su respectiva elipse de dispersión empleando la distancia de Mahalanobis



Fuente: elaboración propia.

```
    return(scores)
}
```

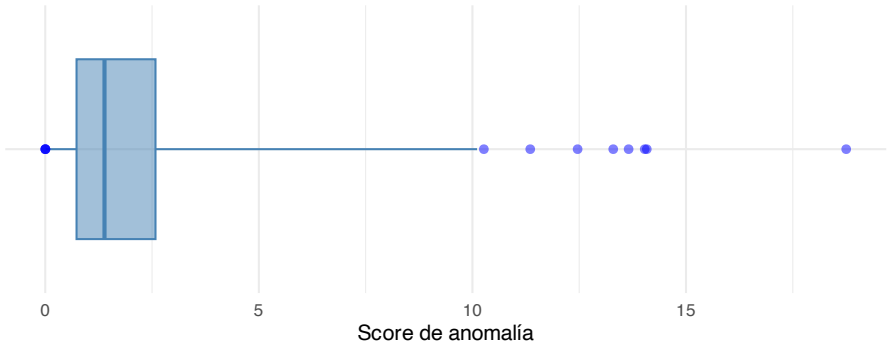
Para nuestro caso, podemos calcular el *score* de anomalía para todas las observaciones empleando el siguiente código:

```
score_PCA <- score.anomal.PCA(X = pca_final, p = 2)
# mirar las 5 observaciones con mayor score de anomalía
head(score_PCA, 5)
```

```
##     Id de la observación Score de anomalía
## 916                  916          18.74997
## 808                  808          14.08165
## 918                  918          14.03348
## 88                    88          13.65659
## 66                    66          13.29652
```

Nota que la observación con el *score* de anomalía más grande es la 916. También podemos visualizar todos los *scores* con un boxplot (ajustado según Hubert y Vandervieren (2008)) como el que se presenta en la Figura 5.7[3].

Figura 5.7. Boxplot ajustado según Hubert y Vandervieren (2008) para los scores de anomalía



Fuente: elaboración propia.

En la Figura 5.7 podemos ver que existen 12 valores atípicos. Es decir, observaciones con *scores* de anomalía relativamente grande.

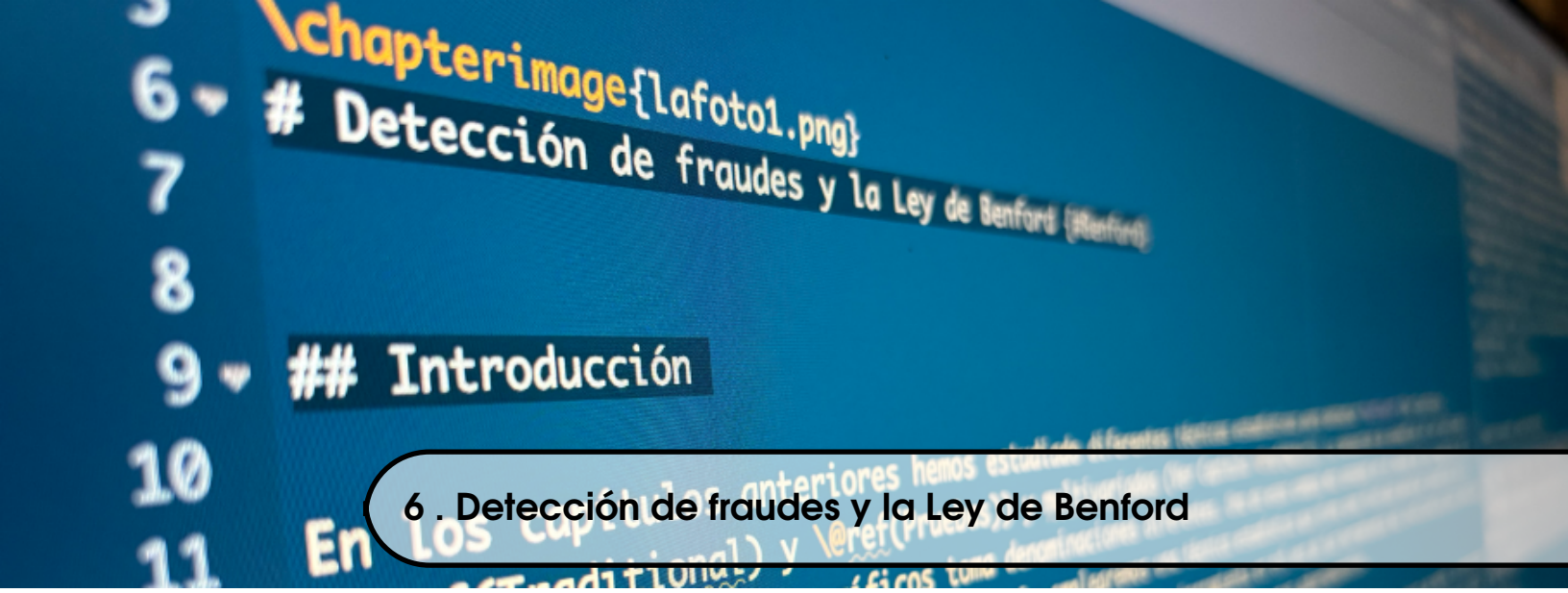
5.6 Comentarios finales

En este capítulo estudiamos cómo emplear la técnica del PCA para detectar *outliers* multivariados. Esta técnica, al igual que las estudiadas en los Capítulos 2, 3 y 4 permite encontrar anomalías que se pueden caracterizar como anomalías globales.Global}

[3]Intencionalmente se omite el código que genera la Figura 5.7. ¡Intenta reproducir esta visualización!

Recordemos que las anomalías globales son observaciones que son atípicas en relación con todo el conjunto de datos. Estas observaciones son inusuales no solo en su entorno inmediato, sino también en comparación con todas las demás observaciones, ya sea que estemos estudiando datos univariados o multivariados. Las anomalías globales suelen ser puntos extremadamente raros o inesperados en el contexto del conjunto de observaciones en su totalidad.

Por otro lado, las anomalías locales son observaciones que son atípicas en relación con su entorno inmediato. Esto significa que, aunque una observación pueda ser considerada normal en el contexto global de todas las observaciones, puede ser considerada una anomalía si se compara con sus vecinos más cercanos. En el Capítulo 8 empezaremos el estudio de herramientas para detectar este tipo de anomalías. Pero antes de pasar a estudiar esas herramientas, tomaremos un desvío en nuestro camino para discutir en el Capítulo 6 que es un fraude y cómo este concepto está relacionado con las anomalías. Así mismo, discutiremos una técnica estadística para detectar fraudes que se conoce como la Ley de Benford.



6 . Detección de fraudes y la Ley de Benford

6.1 Introducción

En los capítulos anteriores hemos estudiado diferentes técnicas estadísticas para detectar *outliers* (Ver Capítulos 2 y 3) y multivariadas (Ver Capítulo 4). La detección de anomalías al ser aplicada a diferentes campos específicos toma denominaciones diferentes. Uno de esos campos más comunes en el mundo de los negocios es la detección de fraudes. En este capítulo emplearemos una técnica estadística que tiene como finalidad detectar anomalías en el primer dígito de los números: la Ley de Benford. Esta herramienta es tal vez de las herramientas más tradicionales para detectar fraudes en el sector financiero, sector salud, los seguros y las agencias gubernamentales.

La detección de fraudes es una de las aplicaciones más comunes de las técnicas de detección de anomalías; de hecho, la detección de fraudes es en sí una disciplina a la cual muchos contadores, estadísticos y científicos de datos dedican su tiempo. En todos los casos, si bien los términos pueden diferir, las técnicas de detección de fraude tienen como finalidad responder a la pregunta: ¿cuál de estas observaciones se comporta de manera diferente a las demás?

En este capítulo, antes de discutir la Ley de Benford, estudiaremos qué se entiende por fraude y los tipos de fraude.

6.2 Detección de fraudes

En general, el fraude se define como un crimen poco común, bien considerado, imperceptiblemente oculto, que evoluciona en el tiempo y, a menudo, cuidadosamente organizado, que aparece en muchos tipos y formas. Empleando herramientas de *business analytics*, se pueden examinar millones de acciones (transacciones) para detectar patrones y detectar acciones (transacciones) fraudulentas. El fraude es (típicamente) muy raro, pero el costo de no detectarlo puede ser enorme. Por ejemplo, según HSN-Consultants (2021) las compañías de tarjetas de crédito perdieron en 2021

a nivel mundial 6.8 centavos de dólar por cada US$100 de transacciones debido a fraude.

Existen muchas herramientas para detectar fraudes en un conjunto de datos. Estas van desde modelos de regresión logística[1], hasta modelos de redes neuronales. Sin importar su origen, todas las herramientas de detección de fraude enfrentan tres grandes retos:

- deben evitar acosar a los buenos clientes.
- deben ser eficientes en su operación. Por ejemplo, en la detección de fraude de tarjetas de crédito se debe tomar una decisión en menos de 8 segundos (HSN-Consultants, 2021).
- debe ser robusto o manejar de alguna manera el desequilibrio que típicamente existe en las muestras. Es decir, típicamente las observaciones con fraude en una muestra son una proporción muy pequeña. Esto le puede generar sesgo a algunas herramientas de analítica. Por ejemplo, el fraude de tarjetas de crédito representa menos del 0.5% de todas las transacciones (HSN-Consultants, 2021).

Toda herramienta de detección de fraudes enfrenta estos tres retos en su tarea de descubrir anomalías. Es decir, detectar eventos que, en comparación con el comportamiento típico, simplemente no "encajan".

Tipos de fraude

En últimas un fraude es un comportamiento anómalo que dependiendo del contexto y negocio se puede clasificar en los siguientes tipos:

- Fraude contra el lavado de dinero (SARLAF)
- Fraude de tarjetas de crédito
- Fraude aduanero
- Falsificación
- Robo de identidad
- Fraude de seguros
- Fraude hipotecario
- Fraude de no entrega
- Fraude en línea
- Fraude de garantía del producto
- Evasión fiscal
- Fraude de telecomunicaciones
- Robo de inventario
- Fraude de entradas
- Fraude electrónico
- Fraude de compensación de trabajadores

Las características más importantes que debe tener un modelo exitoso de detección de fraudes son:

[1] Para una discusión del modelo Logit puedes consultar el capítulo 3 de Alonso y Hoyos (2025).

- Precisión (*Accuracy*). Debe tener un poder de detección y exactitud altos cuando se marcan los casos como sospechosos de fraude.
- Interpretabilidad: Debe ser relativamente fácil de explicar el procedimiento para la "marcación" de casos fraudulentos. En la mayoría de los casos, se requiere cierto nivel de comprensión para que la dirección confíe en el modelo y permita su aplicación. Y en otras ocasiones se requiere poder explicar en ambientes judiciales lo que hace la herramienta.
- Cumplimiento normativo. Debe cumplir toda la normatividad vigente, incluyendo aquellas relacionadas con la privacidad.
- Costo razonable. Debe tener un costo razonable su aplicación y generar un retorno a la inversión también razonable. Debe evitarse aquellos modelos que tienen altos costos y pocos retornos a la organización.
- Complementar los enfoques basados en expertos. Debe ser un complemento a los enfoques clásicos de detección del fraude basados en expertos. Estos enfoques siguen siendo de uso generalizado y representan un buen punto de partida y una herramienta complementaria a los modelos basados en datos. No es una decisión entre humano y algoritmo, sino, por el contrario, debe ser un complemento del trabajo del humano.

6.3 La ley de Benford

Tal vez, una de las herramientas más sencillas y más usadas para encontrar fraudes en registros es la **ley de Benford**, o también conocida como la *ley de los Primeros Dígitos* o *el Fenómeno de los Dígitos Significativos*.

El primer dígito se refiere al dígito más a la izquierda. Por ejemplo, en el número $123{,}241$ el primer dígito es el número 1 y en el número 94 el primer dígito es 9. En la Figura 6.1 se presentan 8 números que serían analizados. En la Figura 6.2 se resaltan los dígitos que serán empleados. La Ley de Benford se aplica únicamente a los primeros dígitos y no a todo el número.

Figura 6.1. Ocho números para el análisis

30,003.594	41,877.25
20,180.351	60,638.94
4,260.459	60,145.96
85,047.202	86,867.98

Fuente: elaboración propia.

De regreso a la Ley de Benford, por un momento pensemos en un encuestador que no quiere salir a realizar el trabajo de campo para recoger las encuestas y, por el con-

Figura 6.2. Primer dígito (resaltado) de los ocho números para el análisis

30,003.594	41,877.25
20,180.351	60,638.94
4,260.459	60,145.96
85,047.202	86,867.98

Fuente: elaboración propia.

trario, se dedica a llenar fraudulentamente las encuestas. Lo "natural" sería inventarse números que iniciaran con un número diferente y posiblemente inventar números que de manera uniforme tengan diferentes valores como primera cifra (ver un ejemplo en la Figura 6.3). Pero la Ley de Benford demuestra que existe una distribución muy diferente en unos datos "legítimamente" generados; es más, se espera que la distribución del primer dígito sea más parecida a la presentada en la Figura 6.4. Así, comparando la distribución de la primera cifra de los datos fraudulentamente generados con la esperada por la Ley de Benford (Figura 6.4) permitirá detectar un posible fraude.

En términos sencillos, esta ley implica que el primer dígito de los números encontrados (registros) en bases de datos de fuentes diversas no muestran una distribución uniforme como intuitivamente se esperaría (Ver Figura 6.3), sino que se organizan de tal manera que el dígito "1" es el más frecuente (30.1 % de los registros), seguido de "2" (17.6 %), "3" (12.5 %), y así sucesivamente hasta "9" (4.6 %) (Ver Figura 6.4). Es decir, se encuentra que aproximadamente el 30 % de los registros deben iniciar por un "1", el 18 % por un "2" y así sucesivamente.

En otras palabras, esta ley establece que en muchos conjuntos de datos numéricos, los números que comienzan con el dígito 1 ocurren con mayor frecuencia que aquellos que comienzan con dígitos más altos. De hecho, según la Ley de Benford, aproximadamente el 30.1 % de los números en un conjunto de datos comienzan con el dígito 1, mientras que solo alrededor del 4.6 % comienzan con el dígito 9.

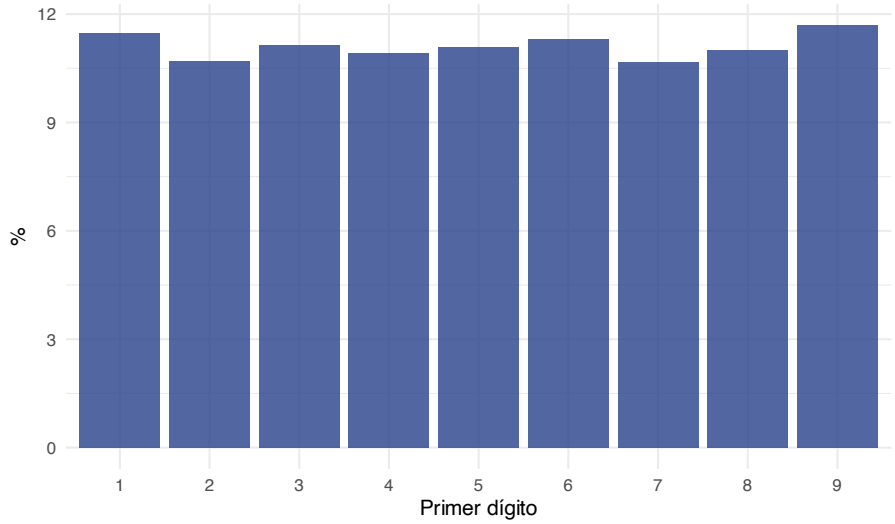
La ley de Benford establece que la probabilidad de observar como primer dígito a d (donde $d = 1, 2, \ldots 9$) será:

$$P(d) = \log_{10}(d+1) - \log_{10}(d) = \log_{10}\left(\frac{d+1}{d}\right) = \log_{10}\left(1+\frac{1}{d}\right) \tag{6.1}$$

Esto implica que la probabilidad de que ocurra cada uno de los 9 dígitos es como la reportada en el Cuadro 6.1.

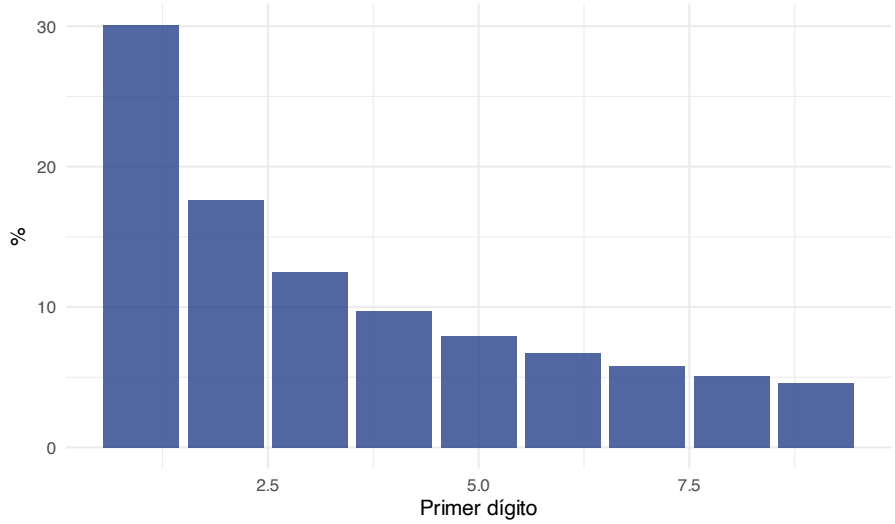
Antes de continuar es importante aclarar que la Ley de Benford no detecta fraude

Figura 6.3. Distribución aproximadamente uniforme del primer dígito



Fuente: cálculos propios.

Figura 6.4. Distribución esperada por la Ley de Benford para el primer dígito



Fuente: cálculos propios.

Cuadro 6.1. Probabilidad de ocurrencia del primer dígito (P(d)) de acuerdo con la Ley de Benford

Digito	y
1	30.10
2	17.61
3	12.49
4	9.69
5	7.92
6	6.69
7	5.80
8	5.12
9	4.58

Fuente: cálculos propios.

en sí misma, sino desviaciones en la distribución de los dígitos que pueden ser consistentes con fraude. Y en este sentido esta ley es considerada una herramienta poderosa frente al fraude, pero no es infalible. Por ejemplo, si un fraude es cometido por alguien que conoce la Ley de Benford y genera datos que cumplen con esta ley, entonces la herramienta no detectará el fraude.

6.4 Un poco de historia

Hablemos un poco de la historia de esta Ley. El primer registro sobre el tema se encuentra en un documento del astrónomo y matemático Simon Newcomb (Newcomb, 1881). La historia cuenta que Newcomb, mientras hojeaba páginas de un libro de tablas logarítmicas, notó que las páginas al principio del libro estaban más sucias que las páginas al final. Esto significaba que sus colegas, que compartían la biblioteca, preferían cantidades que comenzaban con el número uno en sus diversas disciplinas. Él documentó esa regularidad en dos páginas publicadas en el *American Journal of Mathematics* en 1881 (Newcomb, 1881), pero no presentó una demostración de por qué la regularidad debía cumplirse.

En 1938, el físico estadounidense Frank Benford revisó el fenómeno y encontró esa misma regularidad y la llamó la "Ley de Números Anómalos" (Benford, 1938)[2]. Benford publicó en la revista científica *Proceedings of the American Philosophical Society* un estudio en el que empleó 20,000 observaciones de datos recopilados de diversas fuentes para documentar la regularidad. Los datos empleados provenían de diferentes fuentes; desde caudales de ríos hasta pesos moleculares de compuestos químicos. También consideró información de costos, números de direcciones, tamaños de población y constantes físicas. Todos ellos, en mayor o menor medida, siguieron una distribución decreciente de manera exponencial. Este físico no solo documentó la regularidad, sino que también presentó una justificación del porqué ocurre; pero no se

[2]Una reimpresión del artículo original se encuentra en Benford (2012).

presentó una demostración de esta.

Posteriormente, en 1995, el fenómeno fue finalmente comprobado de manera estadística por Theodore P. Hill[3] (Hill, 1995). El profesor Hill presentó una demostración estadística que da sustento al por qué se cumple esta regularidad. Él encontró que si se generan datos de manera legítima que se recolectan en bases de datos y se toman muestras aleatorias de estas, la distribución del primer dígito de la muestra convergerá a la distribución propuesta por Benford.

6.5 Condiciones necesarias para emplear la Ley de Benford

Se ha demostrado que este resultado se aplica a una amplia variedad de conjuntos de datos, incluidas facturas de electricidad, direcciones, precios de acciones, precios de viviendas, números de población, tasas de mortalidad, longitudes de ríos, etc. **En general, una serie de registros numéricos seguirá la ley de Benford cuando esta representa magnitudes de eventos, tales como poblaciones de ciudades, flujos de agua en ríos o tamaños de cuerpos celestes**[4].

Típicamente se espera que la Ley de Benford funcione bien si se cumplen los siguientes cinco supuestos básicos en los datos:

- Se examinará una **variable numérica positiva que no puede tomar el valor de cero**.
- Las observaciones para la variable son **generadas aleatoriamente**, o vienen de un proceso que implica cierto grado de aleatoriedad.
- Los valores que toma la variable **no se encuentran restringidos por un máximo o mínimo**. Es decir, los datos no están censurados.
- Números **no asignados de manera administrativa** (como el de las cédulas o identificaciones o turnos).
- Se cuenta con una **muestra grande**.
- La variable tiene un orden de magnitud relativamente grande (en inglés *Magnitude of order*). El orden de magnitud es algo así como el rango en el que se pueden observar los datos. Por ejemplo, para que la Ley de Benford funcione debemos tener una variable cuyos valores puedan pasar en los valores que toma de 10, 100, 1,000, 10,000, etc. El orden de magnitud típicamente se mide en una escala logarítmica y se emplea para hacer más intuitivo y comprensible el tamaño de los números con los que se trabaja.

Así, estos supuestos implican que la muestra con la que se trabaja en un análisis que emplee la ley de Benford debe ser grande, no tener límites a los valores que puede tomar la variable que será valorada y se generen de forma aleatoria. Por ejemplo, los salarios por hora de operarios en una fábrica muy probablemente tendrán un mínimo y posiblemente algún máximo. Así no tendría mucho sentido emplear la Ley de Benford para detectar un posible fraude en el salario por hora de los operarios. Tampoco

[3] Theodore P. Hill era en ese momento profesor de matemáticas de la Universidad de Georgia Tech en los Estados Unidos.

[4] En el siguiente enlace puedes encontrar diferentes ejemplos en los que funciona la Ley de Benford: https://testingbenfordslaw.com.

tendría sentido analizar los números telefónicos, números de cuentas bancarias o las cédulas digitadas en una base de datos.

Antes de aplicar la ley de Benford, siempre será necesario asegurarnos de que los supuestos se cumplen. En especial, en la práctica, el supuesto más fácil de violar es que los datos sean generados aleatoriamente sin límites.

Ahora, esto puede sonar restrictivo, pero la verdad no lo es. Por ejemplo, variables como facturas a clientes, desembolsos de dinero, pagos recibidos, artículos de inventario típicamente cumplen los supuestos. Algunos autores recomiendan muestras de al menos 100 observaciones, pero para estar más seguros sería razonable emplear muestras de más de 1000 registros.

Para establecer si se cuenta con un orden de magnitud razonable para la aplicación de esta Ley, se cuenta con el indicador de Orden de Magnitud (conducido como el OOM por su sigla en inglés). El OOM se define como:

$$OOM = log_{10}\left(\frac{max\,(X)}{min\,(X)}\right) \tag{6.2}$$

donde X es la variable que se analizará con la Ley de Benford. Para cumplir el supuesto de la ley de Benford, se espera que OOM sea superior a 3 (Ver Kossovsky (2021) para una mayor discusión al respecto). También se ha propuesto una versión del OOM que sea robusta a la presencia de los valores extremos. Esta versión robusta del OOM (ROM) se define como:

$$ROM = log_{10}\left(\frac{P_{99}\,(X)}{P_1\,(X)}\right) \tag{6.3}$$

donde P_{99} y P_1 son el percentil 99 y el primero de la variable bajo estudio, respectivamente. Así, como el caso del OOM, se espera que el ROM sea superior a 3 para tener un orden de magnitud adecuado para aplicar la Ley de Benford. No obstante, algunos autores como Kossovsky (2021) sugieren que incluso valores de ROM de 2.8 o incluso 2.5 podrían ser aceptables para aplicar la Ley de Benford, pero no por debajo de 2.5. No obstante, la regla común es emplear un ROM (o OOM) mayor que 3.

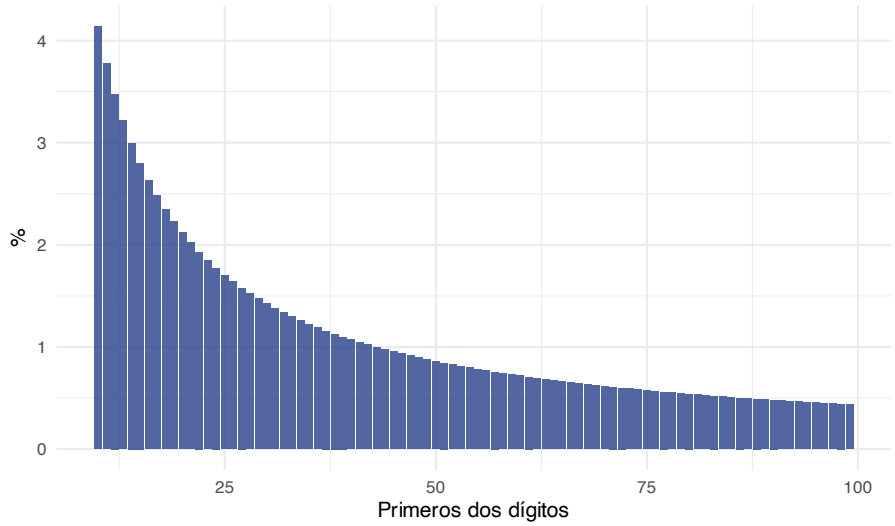
Finalmente, es importante anotar que una variable se ajusta mejor a esta Ley si proviene de una distribución que tiene una media menor que la mediana (sesgo positivo), y los datos no se concentran alrededor de la media. Es decir, para distribuciones no simétricas.

Para pasar a trabajar con un ejemplo, es importante mencionar que, así como existe una distribución para el primer dígito, también existe otra para los dos segundos (ver recuadro), terceros y los demás dígitos. Esa distribución es diferente y corresponde a una generalización de la Ley de Benford. La probabilidad de ocurrencia de los dos primeros dígitos (d_1d_2), es decir la probabilidad de observar en la primera posición el dígito d_1 y en la segunda d_2, está dada por:

$$P(d_1d_2) = \log_{10}\left(1 + \frac{1}{d_1d_2}\right) \tag{6.4}$$

donde $d_1 d_2 \in [10, 11, \ldots 98, 99]$. La probabilidad esperada por la Ley de Benford para los dos primeros dígitos se muestra en la Figura 6.5.

Figura 6.5. Distribución esperada por la Ley de Benford para los dos primeros dígitos



Fuente: Cálculos propios.

6.6 Implementando la Ley de Benford en R

En lo que resta de este capítulo emplearemos la Ley de Benford y R para detectar si existe algún tipo de anomalía o fraude en las facturas que se reportan en los pedidos que se presentan en la base de datos de facturas recibidas por una división de la empresa de servicios públicos *West Coast*. Los datos se encuentran en el archivo `datosBenford.txt` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalias). Estos datos son provistos por Nigrini (2012).

Esta base de datos tiene los registros de 184,412 facturas por pagar por parte de la empresa de servicios públicos *West Coast* a sus proveedores. Los datos corresponden al 2010 y las variables disponibles son:

- `registro`: consecutivo de la factura
- `VendorNum`: número del proveedor
- `Date`: fecha de emisión de la factura
- `Amount`: valor de la factura.

La pregunta de negocio que se desea responder en este caso es: ¿tenemos fraudes en el proceso de facturación de los servicios públicos de *West Coast*? Así, nuestra misión será detectar si existen o no anomalías en el valor facturado (la variable `Amount`).

Empecemos por cargar los datos.

```
# Cargar datos
datos_WC <- read.table("./datos/datosBenford.txt", sep = ";", header = TRUE)
glimpse(datos_WC)
```

```
## Rows: 184,412
## Columns: 4
## $ registro  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
## $ VendorNum <int> 2001, 2001, 2001, 2001, 2001, 2001, 2001, 2001, 2001, 2001, ~
## $ Date      <chr> "2/01/10", "2/01/10", "2/01/10", "2/01/10", "2/01/10", "2/01~
## $ Amount    <dbl> 36.08, 77.80, 34.97, 59.00, 59.56, 50.38, 26.57, 102.17, 25.~
```

Antes de empezar con nuestro análisis, calculemos el OOM y el ROM para determinar si el orden de magnitud de los datos permite aplicar la Ley de Benford. Recuerda que queremos detectar si existen o no anomalías en las facturas por pagar (la variable `Amount`).

Siguiendo la expresión (6.2), el OOM se puede calcular con el siguiente código empleando las funciones **max()** y **min()** de la base de R:

```
# calcular el Orden de Magnitud
OOM <- log10(max(datos_WC$Amount)/min(datos_WC$Amount))
OOM
```

```
## [1] 7.427543
```

Y el ROM se puede construir calculando los percentiles con la función **quantile()** de la base de R. En este caso, siguiendo (6.3) el ROM se puede calcular de la siguiente manera:

```
# calcular el Orden de Magnitud robusto
ROM <- log10(quantile(datos_WC$Amount, 0.99)/quantile(datos_WC$Amount, 0.01))
ROM
```

```
##      99%
## 3.702056
```

Tanto el OOM como el ROM son superiores a 3. Y los otros supuestos parecen razonables en este caso, así podemos proceder con nuestro análisis del cumplimiento o no de la Ley de Benford en las facturas por pagar de la empresa de servicios públicos *West Coast* a sus proveedores en el año 2010.

6.6.1 Análisis gráfico

Para construir un gráfico de barras para la frecuencia (histograma) del primer dígito, podemos emplear el paquete *benford.analysis* (Cinelli, 2018). Para esto debemos primero crear un objeto de clase **Benford** empleando la función **benford()** del paquete *benford.analysis* y posteriormente emplear la función **plot()** del mismo paquete.

La función **benford()** crea un objeto con los resultados del análisis de la Ley de Benford. Sus argumentos son:

benford(data, number.of.digits = 2)

donde:

- **data**: un vector con la variable a la que se le realizará el análisis.
- **number.of.digits**: el número de primeros dígitos para los que se realizará el análisis. Por defecto **number.of.digits = 2**, lo cual implica que se realiza el análisis para los dos primeros dígitos.

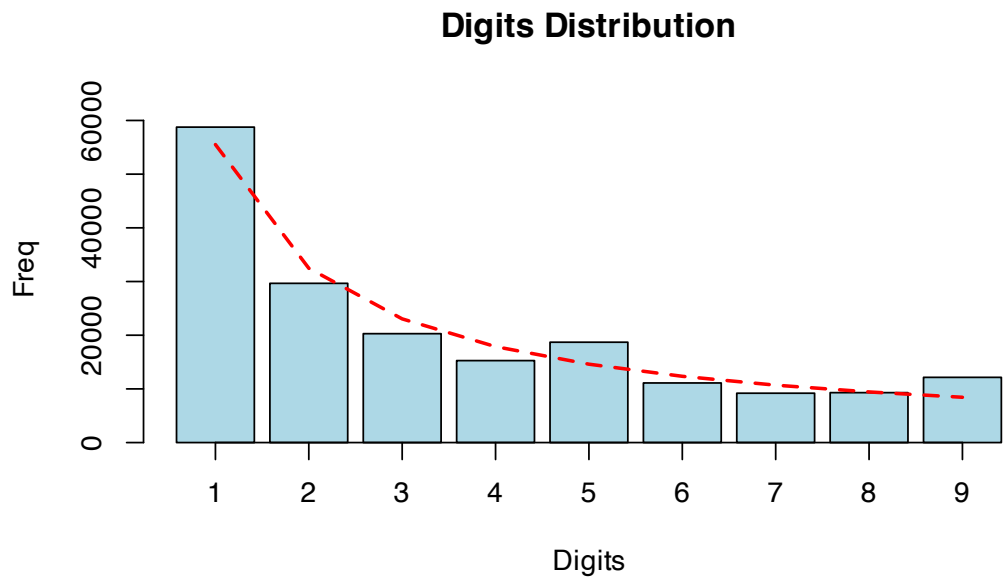
Recuerda que queremos detectar si existen o no anomalías en el valor facturado por pagar (la variable `Amount`). Por tanto, analicemos primero el primer dígito.

```
library(benford.analysis)
# se crea el objeto de clase benford
ben_WC <- benford(datos_WC$Amount, number.of.digits = 1)
```

Ahora grafiquemos el resultado empleando la función **plot()** del paquete *benford.analysis*. Con el argumento **except** excluiremos unas visualizaciones que se producen automáticamente y que no son útiles en este momento para nuestro análisis

```
plot(ben_WC, except = c("second order", "summation", "mantissa", "chi squared",
↪  "abs diff",
    "ex summation", "Legend"), multiple = FALSE)
```

Figura 6.6. Distribución observada (barra) y esperada (línea punteada) del primer dígito para el valor de las facturas por pagar de la empresa West Coast



Fuente: elaboración propia.

En la Figura 6.6 podemos observar que los dígitos 5 y 9 tienen una frecuencia re-

lativamente alta frente a lo esperado. Podemos crear una tabla con las diferencias (absolutas) entre el número de veces que se observa una cifra y el valor esperado por la Ley de Benford empleando la función **suspectsTable()** del paquete *benford.analysis*. Esta función organiza de mayor a menor los dígitos que más se alejan de su valor esperado. Veamos los tres dígitos que más se alejan de lo esperado.

```
head(suspectsTable(ben_WC), 3)
```

```
##    digits absolute.diff
##     <int>         <num>
## 1:      5      4093.028
## 2:      9      3710.770
## 3:      1      3245.456
```

El dígito 5 se aleja en 4093.03 veces lo esperado. Nota que los decimales aparecen porque la predicción de la Ley de Benford no necesariamente será un número entero. En este caso esa diferencia es positiva. Algo similar ocurre para el dígito 9.

Ahora bien, es importante reconocer que es imposible que el número observado de veces de un dígito sea exactamente igual a lo esperado por la Ley de Benford. ¡Estamos en el mundo probabilístico! Por eso será importante realizar pruebas estadísticas que nos permitan determinar si existe una diferencia sistemáticamente significativa entre lo observado y lo esperado por la Ley de Benford.

6.6.2 Análisis estadístico

Para determinar si existe o no una diferencia estadísticamente significativa (con un nivel de confianza de $100 \cdot (1-\alpha)$ %) será necesario emplear pruebas estadísticas.

La primera prueba que consideraremos es de carácter individual. Para determinar si existe una diferencia entre lo observado y lo esperado para cada uno de los dígitos individualmente, podemos crear un intervalo de confianza para la proporción esperada de apariciones de un dígito como primer dígito en la muestra. Si la proporción observada para ese dígito en la muestra está en el intervalo de lo esperado, entonces podremos afirmar con un $100 \cdot (1-\alpha)$ % que la proporción observada no discrepa de lo esperado. En otras palabras, se cumple la Ley de Benford para ese dígito.

Una forma de construir un intervalo de confianza del $100 \cdot (1-\alpha)$ % para la proporción esperada por la Ley de Benford para cada uno de los dígitos (d) es emplear la siguiente fórmula:

$$P\left(d\right) \pm z_{\frac{\alpha}{2}} \cdot \sqrt{n \cdot P(d) \cdot \left(1 - P\left(d\right)\right)} + \frac{1}{2n} \tag{6.5}$$

donde $P(d)$ es la probabilidad esperada por la Ley de Benford de que aparezca el dígito d y n es el número de observaciones.

Esto lo podemos calcular rápidamente con la función **signifd.analysis()** del paquete *BenfordTests* (Joenssen, 2015). Esta función tiene los siguientes argumentos:

signifd.analysis(x = NULL, digits = 1, ci_lines = c(.05))

donde:

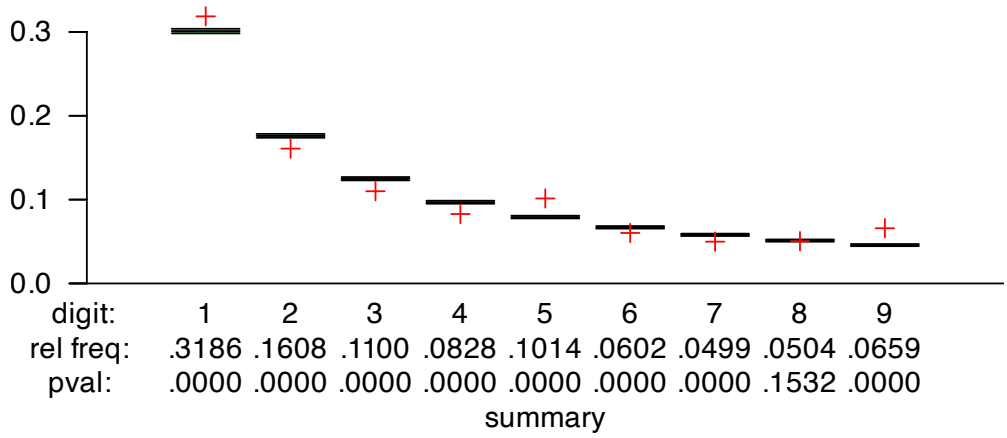
- **x**: es un vector con la variable a la que se le realizará el análisis.
- **number.of.digits**: el número de primeros dígitos para los que se realizará el análisis. Por defecto **digits = 1**, lo cual implica que se realiza el análisis para el primer dígito.

- **ci_lines**: una lista que contiene los niveles de significancia para los intervalos de confianza que se graficarán. Por defecto se graficará un intervalo del 95 % de confianza (**ci_lines = c(.05)**). Nota que se pueden graficar más de un intervalo de confianza.

Empleemos esta función para nuestro ejemplo y construyamos intervalos de confianza con un 99 % de confianza.

```
library(BenfordTests)
# construir intervalos de confianza
signifd.analysis(datos_WC$Amount, digits = 1, ci_lines = c(0.01))
```

Figura 6.7. Intervalos de confianza para la proporción esperada por la Ley de Benford y proporción observada (+) del primer dígito para el valor de las facturas por pagar de la empresa West Coast



Fuente: elaboración propia.

En la Figura 6.7 podemos observar que con excepción del dígito 8, todas las proporciones observadas caen fuera del intervalo esperado por la Ley de Benford. O dicho de otra manera, se puede rechazar la hipótesis[5] que la proporción observada es igual a la predicha por la Ley de Benford, a excepción del dígito 8.

No obstante, estas pruebas individuales pueden estar acumulando muchos errores al estar haciendo muchas comparaciones al mismo tiempo. Sería más conveniente emplear una prueba que probara al mismo tiempo si los datos cumplen o no la Ley de Benford[6].

[5]En este caso podemos emplear los valores p (pval). Recuerda que podemos rechazar la hipótesis nula con un 99 % (95 %) de confianza si el valor p es menor que 0.01 (0.05).

[6]Si estás familiarizado con el análisis estadístico de la regresión múltiple, este es el mismo problema de

La prueba χ^2 de Pearson ($Pearson_{\chi^2}$) permite probar si la distribución de una muestra observada concuerda con una distribución teórica determinada. En este caso, podemos comprobar la hipótesis nula de que las proporciones observadas para cada dígito ($\hat{P}_d$) concuerdan al mismo tiempo con aquellas previstas por la Ley de Benford ($P(d)$) para cada uno de los dígitos ($d = 1, 2 \dots 9$) versus la alterna de que no (no se cumple la Ley de Benford) empleando el siguiente estadístico de prueba:

$$Pearson_{\chi^2} = N \cdot \sum^{d} \frac{\left(\hat{P}_d - P(d)\right)^2}{P(d)} \tag{6.6}$$

Este estadístico de prueba sigue una distribución χ^2 con 8 grados de libertad para el caso del primer dígito y 9 para el de los dos primeros dígitos. Así, en caso de rechazar la hipótesis nula, tendremos evidencia de que los datos no siguen la Ley de Benford.

Este estadístico lo podemos calcular con la función **chisq()** del paquete *benford.analysis* de la siguiente manera:

```
# Prueba Chi cuadrado de Pearson
chisq(ben_WC)
```

```
##
##  Pearson's Chi-squared test
##
## data:  datos_WC$Amount
## X-squared = 4258.6, df = 8, p-value < 2.2e-16
```

En este caso, con un 99 % de confianza, podemos rechazar la hipótesis nula de que la muestra se comporta como lo predice la Ley de Benford.

Otra forma de comprobar el cumplimiento (*complience* en inglés) de la Ley de Benford fue sugerida por Nigrini (2012). Este autor sugiere emplear Desviación Media Absoluta (MAD por la sigla del inglés *Mean Absolute Deviation*) que tiene en cuenta qué tan desviada está cada proporción observada de la teórica sin tener en cuenta el tamaño de la muestra. El MAD está definido como:

$$MAD = \sum^{d} \frac{\left|\hat{P}(d) - P_d\right|}{k} \tag{6.7}$$

donde k es 9 para el caso del primer dígito y 90 para los dos primeros dígitos[7].

Nigrini (2012) provee unos límites para sacar conclusiones con el MAD (Ver Cuadro 6.2).

Este estadístico también lo podemos calcular con el paquete *benford.analysis* por medio de la función **MAD()** .

estar empleando muchas pruebas individuales para sacar conclusiones sobre un conjunto de pendientes. Siempre será mejor una prueba estadística conjunta.

[7] k es el número de posibles combinaciones de dígitos.

Cuadro 6.2. Valores críticos y conclusiones para los valores del MAD

Primer dígito	Dos primeros dígitos	Conlusión (Cumplimiento de la Ley)
0.000 to 0.006	0.000 to 0.012	estrecha
0.006 to 0.012	0.012 to 0.018	aceptable
0.012 to 0.015	0.018 to 0.022	marginalmente aceptable (se cumple para algunos dígitos)
mayor a 0.015	mayor a 0.022	No

Fuente: cálculos propios.

```
MAD(ben_WC)
```

```
## [1] 0.0133147
```

La decisión a partir del Cuadro 6.2 se puede obtener automáticamente con este paquete. De hecho, ya en el objeto `ben_WC` que habíamos calculado con la función **benford()** tenemos este resultado. Observa con cuidado el resultado de llamar este objeto.

```
ben_WC
```

```
##
## Benford object:
##
## Data: datos_WC$Amount
## Number of observations used = 184412
## Number of obs. for second order = 65428
## First digits analysed = 1
##
## Mantissa:
##
##    Statistic   Value
##         Mean  0.4951
##          Var  0.0918
##  Ex.Kurtosis -1.2579
##     Skewness  0.0011
##
##
## The 5 largest deviations:
##
##   digits absolute.diff
## 1      5       4093.03
## 2      9       3710.77
```

```
## 3      1       3245.46
## 4      2       2813.34
## 5      3       2756.20
##
## Stats:
##
##  Pearson's Chi-squared test
##
## data:  datos_WC$Amount
## X-squared = 4258.6, df = 8, p-value < 2.2e-16
##
##
##  Mantissa Arc Test
##
## data:  datos_WC$Amount
## L2 = 0.0040803, df = 2, p-value < 2.2e-16
##
## Mean Absolute Deviation (MAD): 0.0133147
## MAD Conformity - Nigrini (2012): Marginally acceptable conformity
## Distortion Factor: -1.065467
##
## Remember: Real data will never conform perfectly to Benford's Law. You should not focus on p-va
```

La conclusión la muestra automáticamente la función. En este caso, la conclusión es *"MAD Conformity - Nigrini (2012): Marginally acceptable conformity"*. Es decir, marginalmente aceptable, lo cual implica que algunos dígitos se cumplen la Ley de Benford y otros no.

Esto quiere decir que los datos de las facturas por pagar a sus proveedores de la empresa de servicios públicos *West Coast* en el año 2010 no cumplen la Ley de Benford. Estamos frente a la posibilidad de un fraude.

6.6.3 Trabajando con los casos problemáticos.

Cuando se detecta un posible fraude, será necesario buscar dónde está presente. Ya vimos cómo descubrir aquellos dígitos con la mayor desviación a lo esperado por la Ley de Benford con la función **suspectsTable()** . En nuestro ejemplo esos dígitos son el 5 y el 9. Esto quiere decir que las facturas que inician con estos dígitos son las principales sospechosas de fraude.

Esto implicará que el siguiente paso será sacar de la base de datos aquellas facturas cuyos montos inicien con 5 y 9 para un análisis más minucioso. Eso lo podemos hacer con la función **getDigits()** del paquete *benford.analysis*. Necesitamos tres argumentos. El objeto de clase **benford**, la base de datos original y qué (primer) dígito queremos extraer. Por ejemplo, guardemos en un objeto todos los registros (las observaciones) que inician con los dígitos 5 y 9.

```
# Extraer registros con primer dígito 5 y 9
digitos_5_9 <- getDigits(ben_WC, datos_WC, digits = c(5, 9))
head(digitos_5_9)
```

```
##    registro VendorNum    Date Amount
##       <int>     <int>  <char>  <num>
## 1:        4      2001 2/01/10  59.00
## 2:        5      2001 2/01/10  59.56
## 3:        6      2001 2/01/10  50.38
## 4:       12      2001 2/01/10  55.22
## 5:       15      2001 2/01/10  55.29
## 6:       19      2001 2/01/10  94.29
```

Con este objeto de clase **data.frame** ya se puede empezar el análisis de las facturas sospechosas en otras áreas de la organización.

Otra tarea que nos gustaría hacer es encontrar los duplicados. Explora las funciones **duplicatesTable()** y **getDuplicates()** del paquete *benford.analysis*. Por ejemplo,

```
# imprimir los duplicados por orden decreciente
duplicatesTable(ben_WC)

# obtener las observaciones de los 2 valores con más duplicados
duplicados <- getDuplicates(ben_WC, datos_WC, how.many = 2)
```

Por otro lado, no consideramos el análisis de los dos primeros dígitos. Con lo estudiado tú puedes constatar que tampoco se cumple la ley de Benford para los dos primeros dígitos. Esto podría ayudar a acotar la búsqueda de los fraudes. ¡Realiza este análisis!

6.7 Comentarios Finales

En este capítulo hemos definido lo que se entiende por fraude y cómo la tarea de detección de fraudes es un caso especial de la detección de anomalías. También estudiamos la Ley de Benford, que es una herramienta estadística para encontrar anomalías univariadas. Específicamente, estudiamos cómo encontrar anomalías relacionadas con el primer dígito, pero es muy fácil generalizar lo estudiado para los primeros dos dígitos.

La Ley de Benford ofrece una aproximación estadística sencilla y poderosa para detectar irregularidades en datos, especialmente los contables y financieros. No obstante, debe complementarse con otras técnicas de auditoría y de análisis de anomalías, pues por sí sola no constituye una prueba concluyente de fraude.

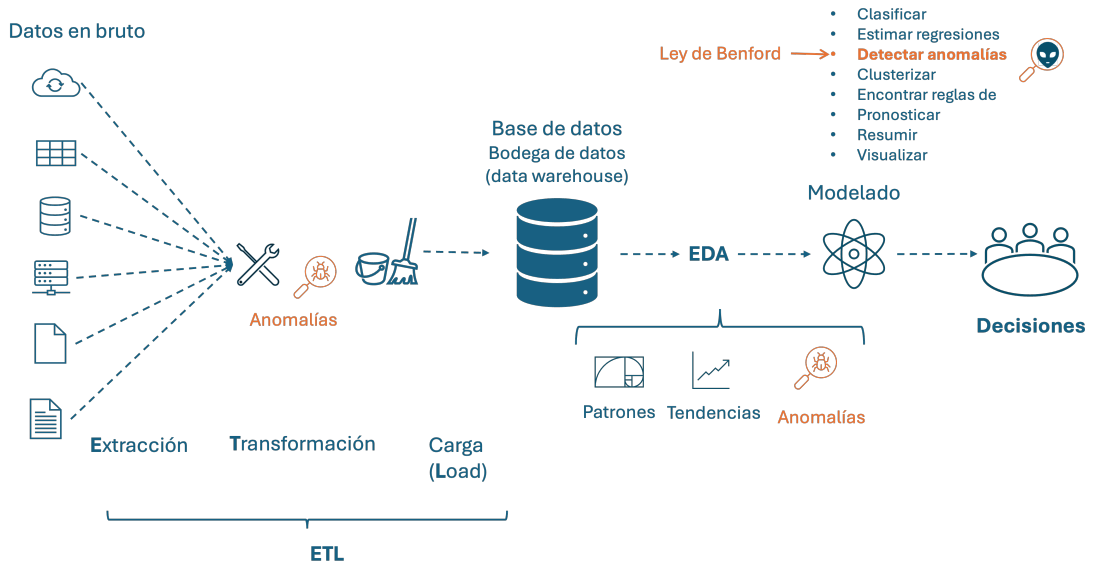
También es importante anotar que el cumplimiento de esta Ley no necesariamente significa que no existe fraude. Significa que no se adulteraron las cifras, no se generaron de manera artificial. ¡Pero el fraude puede estar presente de muchas otras maneras!

Por eso es común que se combinen diferentes aproximaciones de detección de anomalías en la búsqueda de fraude.

Hasta el capítulo anterior habíamos discutido técnicas de detección de anomalías que son especialmente útiles en los procesos de **ETL** y **EDA**. En el **ETL** común el uso de técnicas univariadas para la detección de *outliers*, mientras que en el **EDA** el análisis de *outliers* multivariados es más común. La técnica que estudiamos en este capítulo no se emplea típicamente en los procesos de **ETL** ni de **EDA**. La ley de Benford es una herramienta que es empleada típicamente en el moldeamiento como tal de los datos, que están directamente relacionados con responder una pregunta específica de negocio.

Si recordamos la ilustración presentada en la introducción del Capítulo 3 (Ver Figura 6.8), la Ley de Benford estaría ubicada en la parte superior derecha. En los Capítulos 9 y 10 estaremos estudiando herramientas, que al igual que la Ley de Benford, son más empleadas en el modelado necesario para responder preguntas específicas de negocio. Pero antes, en el Capítulo 8 estudiaremos unas técnicas del aprendizaje de máquina para detectar anomalías que hacen parte de las herramientas empleadas en el **EDA**.

Figura 6.8. La Ley de Benford como herramienta de modelado en el mundo del business analytics



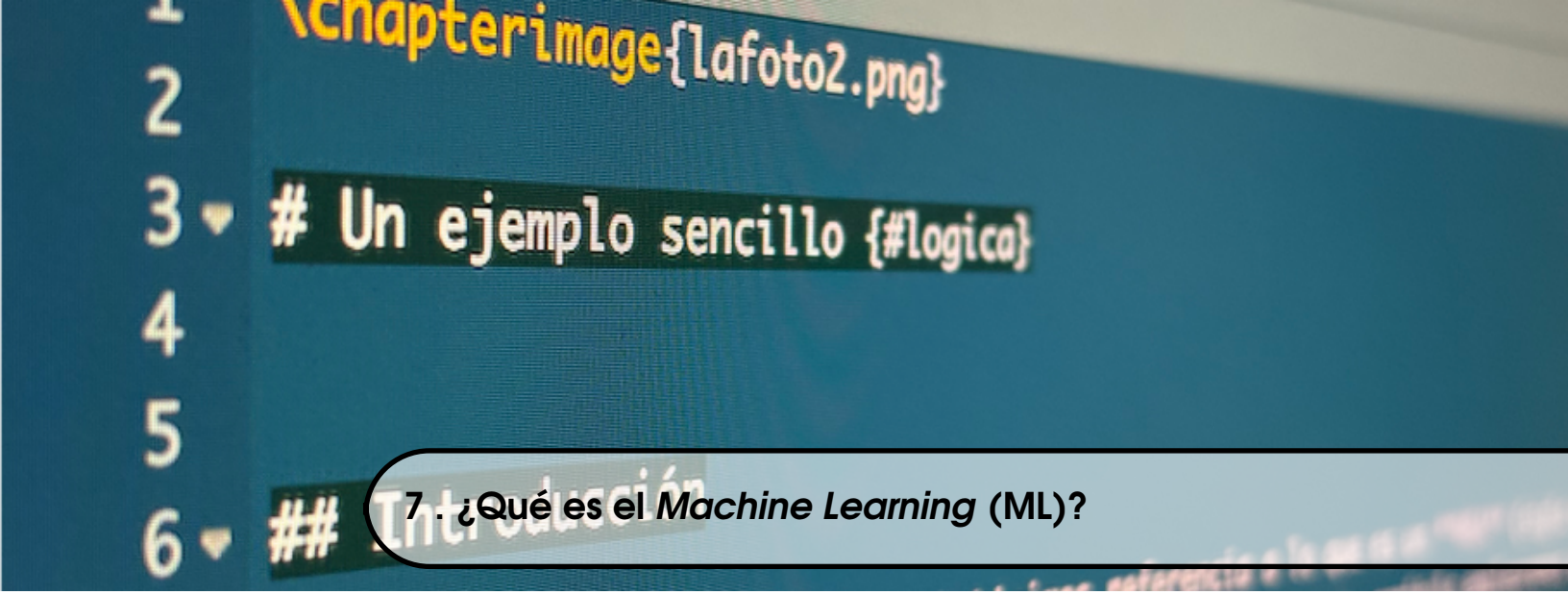
Fuente: elaboración propia.

Con este capítulo se concluye la primera parte del libro, que está dedicada a las técnicas estadísticas para la detección de anomalías. En la tercera parte del libro estudiaremos técnicas de aprendizaje de máquina o **ML** (por sus siglas en inglés de *Machine Learning*) para la detección de anomalías. En el Capítulo 8 estudiaremos técnicas

basadas en la distancia entre observaciones, en el Capítulo 9 estudiaremos técnicas basadas en la densidad de las observaciones y en el Capítulo 10 estudiaremos técnicas basadas en árboles de decisión.

Parte III

Métodos de origen en el machine learning no supervisado



7 ¿Qué es el *Machine Learning* (ML)?

En los Capítulos 2 al 6 nos concentramos en métodos estadísticos para la detección de anomalías. Estos modelos, desarrollados por la estadística, suelen estar fundamentados en una teoría probabilística explícita y en supuestos formales que permiten hacer inferencia sobre parámetros poblacionales. Ahora damos un paso distinto, en la tercera parte de este libro (**Métodos de origen en el machine learning no supervisado**) exploraremos métodos cuya raíz proviene del campo conocido como *Machine Learning* (ML).

El término ML (aprendizaje de máquina en español[1]) hace referencia a un conjunto de algoritmos que, a partir de datos, "aprenden" patrones sin necesidad de estar guiados por un modelo teórico previo o supuestos sobre la población de la que provienen los datos. La diferencia con la aproximación estadística tradicional es sutil pero fundamental. Mientras la estadística clásica busca principalmente explicar y validar hipótesis sobre relaciones poblacionales entre variables. Es decir, se centra en la **inferencia** sobre lo que ocurre en una población, el ML se preocupa por la **capacidad de predicción y generalización** del algoritmo frente a nuevos datos.

Otra diferencia está en la forma de trabajar con los supuestos. En la estadística, el punto de partida son supuestos sobre la distribución de los datos (normalidad, homocedasticidad, independencia, etc.). En este mundo la validez de los resultados depende que los supuestos de los modelos empleados sean validados. En cambio, en el ML los supuestos son menos estrictos y, en muchos casos, el enfoque es más empírico: se evalúa el rendimiento del modelo en función de su capacidad para predecir o clasificar correctamente nuevos datos, sin necesidad de validar formalmente los supuestos estadísticos. En el ML el énfasis está en la **práctica** y en la **utilidad** del modelo, más que en su fundamentación teórica o la capacidad de hacer inferencia sobre la población de la que provienen los datos. En este mundo la inferencia sobre el comportamiento poblacional no es el objetivo principal. Y por eso en este mundo los resultados típicamente

[1] En la jerga del *business anallytics* es más común que el término se emplee en inglés y no en español. En este libro seguiremos esa tradición de este campo del conocimiento.

no implican intervalos de confianza o pruebas de hipótesis.

Antes de iniciar esta sección del libro, es importante aclarar la relación de conceptos como el *machine learning* y la **inteligencia artificial** (IA). La IA es un campo amplio de las ciencias de la computación que desarrolla sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el razonamiento, el aprendizaje, la percepción y la toma de decisiones. Dentro de la IA se encuentran áreas no mutuamente excluyentes como *machine learning*, *deep learning*, procesamiento de lenguaje natural, visión por computador, robótica, planificación y sistemas expertos (Mukhamediev et al., 2022).

El área más relevante de la IA para el *business analytics* es el *machine learning*, entendido como el conjunto de algoritmos que aprenden de los datos y generalizan a nuevos casos. Dentro de este se encuentra el *deep learning*, que utiliza redes neuronales profundas para identificar patrones complejos en grandes volúmenes de datos. Los modelos de lenguaje a gran escala (LLM), como GPT o Gemini, constituyen ejemplos recientes de *deep learning* aplicados al procesamiento del lenguaje natural, capaces de generar texto coherente y contextualizado (Bommasani et al., 2022). En Alonso y Serrano (2025) puedes encontrar una mayor discusión sobre estos temas.

El ML es parte de la IA, pero no toda la IA es ML (Alonso y Serrano, 2025). Mientras que la IA también incluye métodos simbólicos y enfoques basados en reglas, el ML representa el componente que aprende directamente de los datos. La Figura 7.1 resume esta jerarquía conceptual de manera esquemática. El ML como subconjunto dentro del amplio dominio de la IA[2].

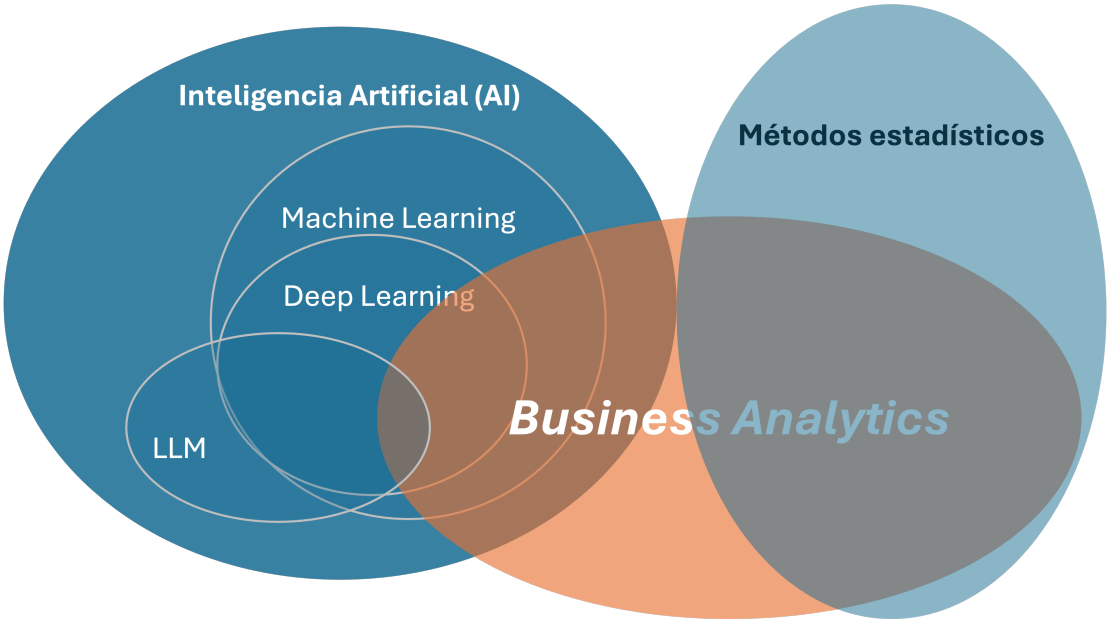
En el ML se distinguen dos grandes familias de algoritmos: **supervisados** y **no supervisados**. En el aprendizaje supervisado, los datos contienen una variable objetivo conocida (etiqueta) y el modelo aprende a predecirla. Por ejemplo la tarea de clasificación o la regresión son ejemplos de aprendizaje supervisado. En la detección de anomalías, los métodos supervisados se emplean cuando se cuenta con un conjunto de datos etiquetados que identifican cuáles observaciones son normales y cuáles son anomalías. El modelo aprende a distinguir entre ambas clases basándose en las características de los datos. Sin embargo, en la práctica, este tipo de datos etiquetados son escasos o inexistentes, lo que limita la aplicabilidad de los métodos supervisados para la detección de anomalías.

En el aprendizaje no supervisado, en cambio, no existe una etiqueta previa. El objetivo es descubrir estructuras ocultas en los datos, como agrupaciones de individuos (*clústering*) o la identificación de observaciones atípicas (detección de anomalías). En la detección de anomalías, los métodos no supervisados son especialmente útiles cuando no se dispone de etiquetas que identifiquen las anomalías. El modelo busca patrones en los datos y señala aquellas observaciones que se desvían significativamente de estos patrones como posibles anomalías. Este es el caso de los ejemplos que hemos estudiado hasta ahora en este libro y los que estudiaremos en los siguientes capítulos.

La Figura 7.1 también ilustra la relación entre la estadística y el *business analytics*. El

[2]Ver Alonso y Serrano (2025) para una mayor discusión del tema.

Figura 7.1. Relación entre la IA, la estadística y el business analytics



Fuente: tomado de Alonso y Serrano (2025).

business analytics es un campo interdisciplinario que combina estadística, ML, gestión empresarial y toma de decisiones basada en datos para resolver problemas empresariales. En conjunto, estas disciplinas permiten a las organizaciones extraer valor de sus datos para mejorar la eficiencia operativa, optimizar estrategias y tomar decisiones informadas (Alonso y Serrano, 2025). En los Capítulos 2, 3, 4, 5 y 6 nos concentramos en métodos estadísticos para la detección de anomalías.

En esta parte del libro (Capítulos 8, 9 y 10) nos concentraremos en métodos de origen en el ML, particularmente en los métodos no supervisados, que como se mencionó en el Capítulo 1 son los tipos de modelos en los que nos concentramos en este libro ya que la mayoría de herramientas para detectar anomalías en el mundo del aprendizaje supervisado corresponden a modelos de clasificación que puedes estudiar en profundidad en Alonso y Hoyos (2025).

En el Capítulo 8 estudiaremos los métodos basados en la distancia entre observaciones, particularmente el método de los k vecinos más cercanos (**kNN**). En el Capítulo 9 estudiaremos los métodos basados en densidad, particularmente el método **DBSCAN** y sus variantes. Finalmente, en el Capítulo 10 estudiaremos los métodos basados en árboles de decisión, particularmente el método *Isolation Forest*. Con el estudios de estas aproximaciones podemos completar nuestra caja de herramientas para la detección de anomalías en datos no etiquetados como anómalos.

8 . Métricas de origen en el ML: distancia kNN y LOF

En los Capítulos 2, 3 y 6 estudiamos métodos de origen en la estadística para detectar anomalías puntuales y univariadas. En el Capítulo 4 discutimos cómo encontrar *outliers* multivariados puntuales. Como lo discutimos en el Capítulo 7, todas las métricas y pruebas empleadas hasta aquí tienen en común su origen en la estadística.

En este capítulo estudiaremos dos métricas cuyo origen se encuentra en el aprendizaje de máquina: la distancia **kNN** y el **LOF** (por la sigla del inglés *Local Outlier Factor*). Ambas son técnicas para detectar anomalías multivariadas puntuales o globales que tienen origen en métodos de aprendizaje de máquina. La diferencia entre estas aproximaciones es la filosofía de cómo encontrar las anomalías.

La distancia **kNN**, como su nombre lo indica, es un método basado en la distancia como aquellos estudiados en los Capítulos 2, 3 y 4. Intuitivamente, la distancia **kNN** busca encontrar observaciones que estén "lejos" de sus vecinos más próximos. Por otro lado, el **LOF** es un método basado en la densidad. Intuitivamente, el **LOF** busca encontrar observaciones que se encuentren en regiones de baja densidad de datos. En otras palabras, el **LOF** busca encontrar observaciones que estén alejadas de sus vecinos, pero cuyos vecinos estén "densamente" agrupados. En las secciones de este capítulo discutiremos cada una de estas métricas en detalle y cómo implementarlas en R.

8.1 Distancia kNN

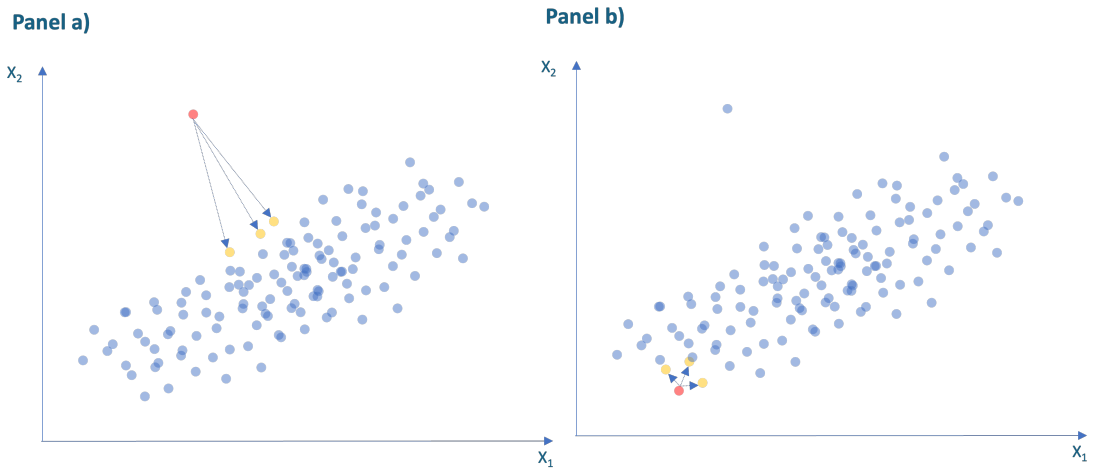
Si te son familiares los modelos de clasificación, debes conocer el modelo de k vecinos más próximos (**kNN** por sus siglas en inglés *k-nearest neighbors*)[1]. Intuitivamente, el modelo **kNN** clasifica una observación basándose en la clase mayoritaria de sus k vecinos más cercanos. Por ejemplo, si $k = 5$[2] y 3 de los vecinos más cercanos a una

[1] Puedes ver Alonso y Hoyos (2025) para una introducción a los modelos de clasificación.
[2] Es decir, se consideran 5 vecinos para clasificar cada observación.

observación pertenecen a la clase A y 2 a la clase B, entonces la observación se clasifica como A. La distancia **kNN** que emplearemos en este capítulo es un concepto que se deriva de dicho modelo.

La distancia **kNN** mide la distancia promedio de un individuo a cada uno de los *k* individuos más cercanos. En la Figura 8.1 se ilustran las distancias para los tres vecinos más cercanos ($k = 3$) de la observación en rojo. Las distancias individuales están representadas por las flechas azules y los tres vecinos más cercanos que se presentan en amarillo. En el panel a) se observa un punto con distancia promedio a los vecinos (**kNN**) relativamente grande y por tanto se considerará como una anomalía. Por otro lado, en el panel b) se presenta una observación con una distancia **kNN** (promedio)) relativamente pequeña. De esta manera, la distancia **kNN** proporciona una medida intuitiva de qué tan aislada está un individuo de sus vecinos, de tal manera que los valores más grandes son más probables de estar asociados a anomalías.

Figura 8.1. Relación entre las tareas de analítica y los tipos de analítica



Fuente: elaboración propia.

Ramaswamy et al. (2000) propuso emplear la distancia **kNN** para detectar las observaciones que se encuentran "más alejadas" de las otras. En general, los pasos para encontrar la distancia **kNN** son:

1. Estandarizar[3] los datos.
2. Seleccionar una observación y calcular la distancia de esta a cada una de las demás observaciones.
3. Identificar los k vecinos más cercanos a la observación seleccionada.
4. Calcular la distancia promedio del punto a sus k vecinos más cercanos.
5. Repetir pasos 2 a 4 hasta agotar la muestra.

[3]Es decir, se emplean variables centradas (se resta la respectiva media) y con varianza igual (se divide cada una por la respectiva desviación estándar) para evitar que las escalas de medición de las variables y su volatilidad afecten el resultado, a esto lo llamaremos estandarizar los datos.

La distancia entre cada observación se puede calcular de diferentes maneras; por ejemplo, empleando la distancia euclidiana. Pero existen más tipos de distancias, como lo discutimos en la Sección 4.2.

La **distancia euclidiana** entre dos individuos se define como la distancia al cuadrado entre los dos vectores[4]. En otras palabras, se calcula la distancia cuadrada entre los dos individuos para cada una de las d dimensiones (variables), posteriormente se suman esas distancias y finalmente se regresan a la escala original calculando la raíz cuadrada[5].

Formalmente, definamos el vector de las observaciones[6] para las d variables para un individuo i, como $\mathbf{x_i} = [x_{i,1} \quad x_{2,1} \quad \cdots \quad x_{i,d}]$. Entonces, la **distancia euclidiana** se define como:

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\left(\sum_{m=1}^{d} (x_{i,m} - x_{j,m})^2\right)} = \sqrt{(\mathbf{x_i} - \mathbf{x_j})^T (\mathbf{x_i} - \mathbf{x_j})} \tag{8.1}$$

La distancia euclidiana es la medida de distancia más común, pero tiene un problema; a medida que aumenta la cantidad de variables involucradas en el cálculo de las distancias (dimensionalidad de los datos), la distancia euclidiana resulta menos útil[7]. Existen otras medidas como la de Chebyshev (o Distancia máxima), Manhattan, Canberra, Minkowski y binaria[8].

La **distancia de Chebyshev Distancia máxima** o también conocida como la norma vectorial máxima se define como:

$$d(\mathbf{x_i}, \mathbf{x_j}) = \max_{1 \leq m \leq d} |x_{i,m} - x_{j,m}| \tag{8.2}$$

La **distancia de Manhattan** también conocida como la distancia del taxi (en inglés *Taxicab distance*) o distancia de la manzana (en inglés *City Block distance*) está definida como:

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sum_{m=1}^{d} |x_{i,m} - x_{j,m}| \tag{8.3}$$

La **distancia de Canberra** es una versión ponderada de la distancia Manhattan que se define como:

[4] Esta es la norma vectorial de orden 2.

[5] Esta es la razón por la que los datos originales deben ser centrados, como se discutirá más adelante. Es decir, se emplean variables centradas (se resta la media) y escaladas (divididas por su desviación estándar) para evitar que las escalas de medición de las variables afecten el resultado, a esto lo llamaremos estandarizar los datos.

[6] Como se mencionó anteriormente, típicamente estas variables estarían centradas en un ejercicio de detección de anomalías.

[7] Esto se conoce como la "maldición de la dimensionalidad".

[8] Ver Sección 2.2 de Alonso et al. (2025) para una discusión de esas medidas de distancia.

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sum_{m=1}^{d} \frac{|x_{i,m} - x_{j,m}|}{|x_{i,m}| + |x_{j,m}|} \tag{8.4}$$

La **distancia de Minkowski** es una generalización de las tres primeras distancias anteriores. Esta se define como la norma vectorial de orden p. Es decir, la distancia de Minkowski es la p-ésima raíz de la suma de las p-ésima potencia de las diferencias de los componentes. En otras palabras,

$$d(\mathbf{x_i}, \mathbf{x_j}) = \left(\sum_{m=1}^{d} (x_{i,m} - x_{j,m})^p \right)^{\frac{1}{p}} \tag{8.5}$$

Nota que cuando $p = 1$, entonces esta distancia es exactamente igual a la de Manhattan. Con $p = 2$ esta distancia es igual a la euclidiana. Y finalmente, cuando $p = \infty$ entonces obtendremos la distancia de Chebyshev.

Así, el primer paso para calcular la distancia **kNN** es estandarizar los datos. El segundo paso implica seleccionar una observación y calcularle a esa observación la distancia a todas las otras observaciones empleando alguna de las distancias discutidas anteriormente. Después, el tercer paso implica encontrar para esa observación seleccionada cuáles son las k observaciones que tienen la distancia más pequeña a ella; es decir, los k vecinos más cercanos. El cuarto paso es calcular la distancia promedio a la observación seleccionada de los k vecinos cercanos. Y finalmente, este procedimiento se repite para todas las observaciones de la muestra.

De esta manera, la distancia **kNN** se emplea para encontrar aquellas observaciones que tengan la distancia **kNN** más grande (la distancia promedio a sus k vecinos cercanos) de tal manera que se puede considerar como anomalía. Si el punto de datos se encuentra dentro del rango esperado de los k vecinos más cercanos, se considera un punto "normal" (no anómalo). Lastimosamente, no existe un umbral para determinar cuándo una distancia **kNN** se considera "grande". La evaluación de si una distancia es grande o no dependerá del contexto.

Por otro lado, nota que para emplear este algoritmo, es necesario definir k (el número de vecinos cercanos). Está documentado que la elección del valor de k puede afectar sustancialmente los resultados de la detección de anomalías. Un valor de k demasiado pequeño puede conducir a la detección de falsos positivos, mientras que un valor demasiado grande puede omitir anomalías reales. En la práctica, se recomienda probar diferentes valores de k y evaluar la estabilidad de los resultados.

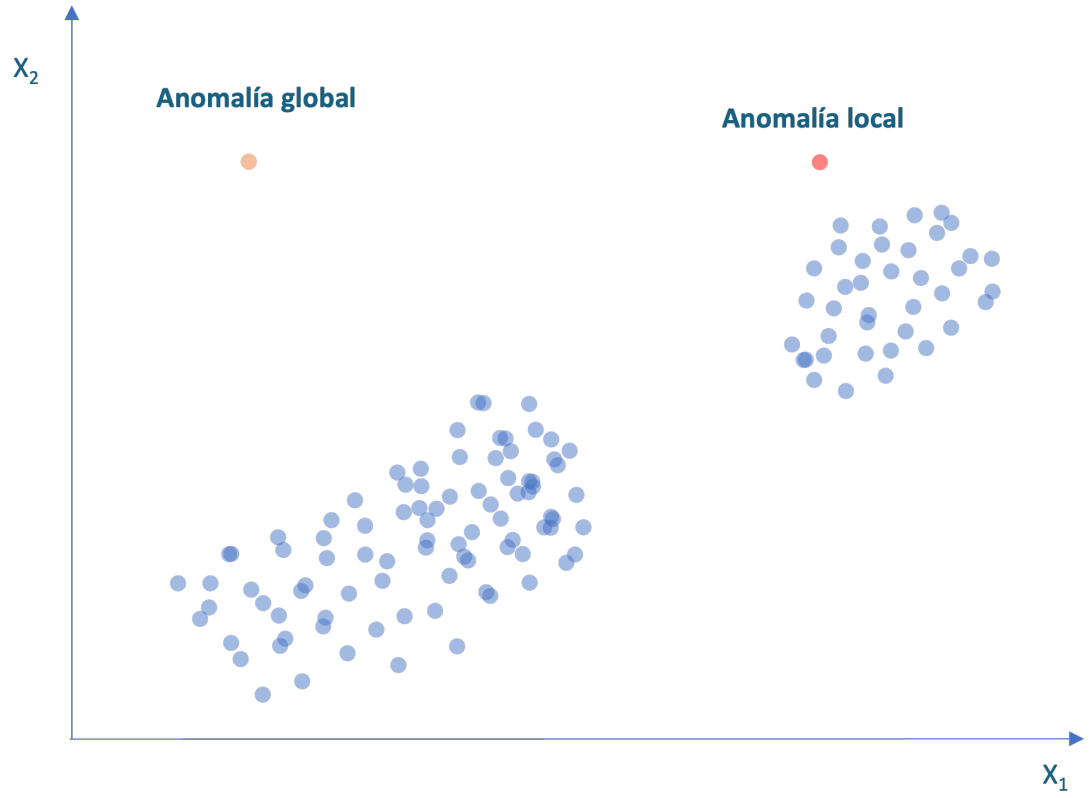
8.2 LOF

El **LOF** (*Local Outlier Factor*, que sería algo así como factor de "atípicidad" local) es una técnica para detectar anomalías propuesta por Breunig et al. (2000) que, a diferencia de métodos basados en la distancia como la distancia **kNN**, busca encontrar la

densidad local para cada observación. Y empleando dicha densidad identifica como anomalías aquellos puntos que se encuentran en **regiones de baja densidad**.

La distancia **kNN** es una métrica buena para detectar puntos que están realmente lejos de sus vecinos (anomalías globales). Por ejemplo, en la Figura 8.2 el punto naranja (situado en la parte superior izquierda) tendrá la distancia **kNN** más grande y por tanto será marcado como una anomalía. No obstante, un punto como el rojo (parte superior derecha) no es detectado como una anomalía por la distancia **kNN** porque está relativamente cerca de sus vecinos más cercanos y, por tanto, tiene una distancia **kNN baja**. Sin embargo, esa observación se encuentra alejada de sus vecinos más próximos, que están "densamente" agrupados. Este tipo de observaciones se denominan anomalías locales y se identifican más fácilmente con el **LOF**.

Figura 8.2. Anomalía global y local



Fuente: elaboración propia.

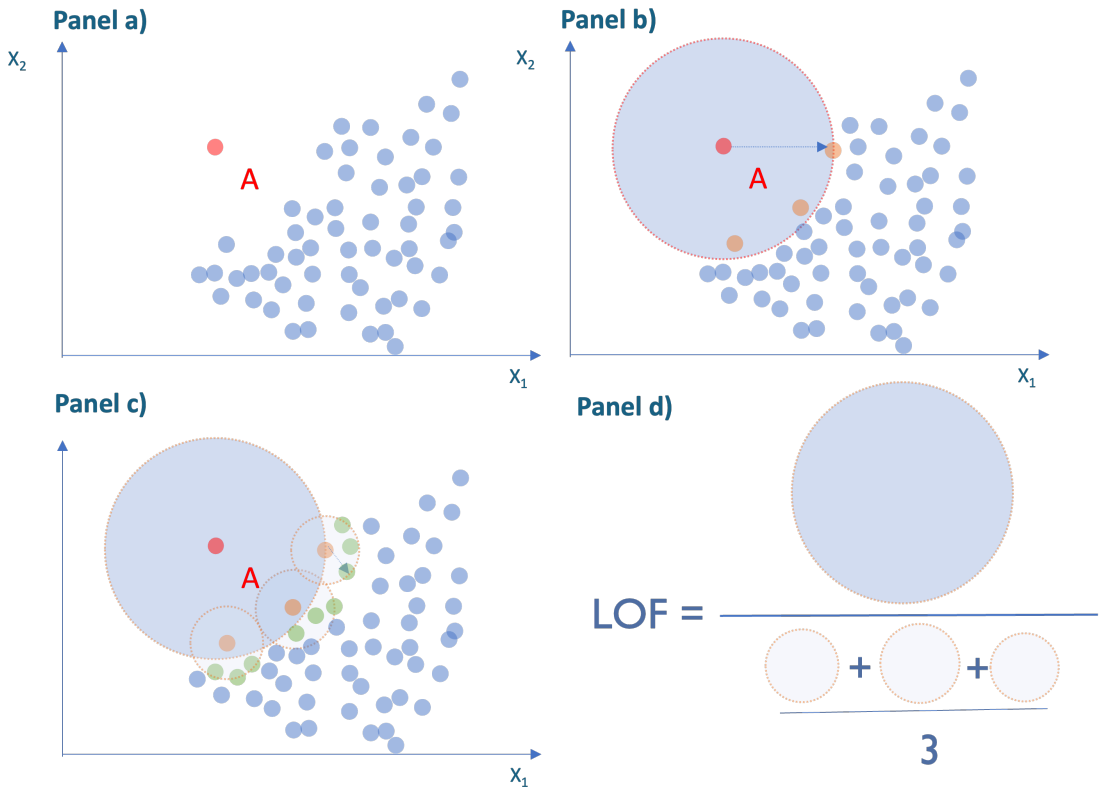
El **LOF** para una observación se define como la densidad promedio alrededor de los k vecinos más próximos del individuo dividida por la densidad alrededor del propio individuo.

Antes de estudiar la formalidad del **LOF**, veamos un poco la intuición detrás de este

factor. Consideremos una observación como el punto A en rojo en el panel a) de la Figura 8.3 y supongamos que $k = 3$.

El primer paso es considerar los 3 vecinos más cercanos a esta observación (Ver observaciones en naranja en el panel b) de la Figura 8.3). Para determinar la densidad del punto A, calculamos la distancia de cada vecino a la observación analizada. Y en vez de sacar el promedio de las distancias de los vecinos, como se hace en el método de la distancia **kNN**, consideramos la distancia más grande y esta se convierte en el radio del área para establecer la densidad del punto A. Noten que entre más grande es el radio, más baja es la densidad de puntos al rededor del punto bajo observación.

Figura 8.3. Ejemplo del cálculo del LOF



Fuente: elaboración propia.

Ahora realizamos el mismo ejercicio de establecer la densidad para cada uno de los vecinos de A (Ver panel c) de la Figura 8.3). Finalmente, el **LOF** para el individuo A es la razón entre el área de la densidad de A y el promedio de sus vecinos. Nota que en este caso el **LOF** será mayor que uno, lo cual es un indicador de que probablemente es una anomalía; más adelante discutiremos esto en detalle.

Recordemos que hemos definido $d(x_i, x_j)$ como la distancia entre las observaciones

i-ésima y $j$ésima. Además, sean:

- $d_{i,k}$: la distancia entre la observación i-ésima y k-ésimo vecino más próximo.
- $N_{i,k}$: el conjunto de k vecinos más próximos dentro de $d_{i,k}$ de[9] x_i.
- $r_k(x_i, x_j) = \max(d_{i,k}, d(x_i, x_j))$: es el "alcance" (en inglés *reachability*) de x_i desde x_j. Es decir, $r_k(x_i, x_j)$ es la distancia entre las observaciones x_i y x_j cuando están alejados entre sí, pero es igual a $d_{i,k}$ si x_i es uno de los k vecinos más cercanos de[10] x_j.

Siguiendo a Breunig et al. (2000), podemos definir la densidad de "alcance" local (en inglés *local reachability density*) de la observación x_i ($lrd_k(x_i)$) como[11]:

$$lrd_k(x_i) = \frac{1}{\frac{\sum_{x_j \in N_{i,k}} r_k(x_i, x_j)}{N_{i,k}}} \tag{8.6}$$

El **LOF** corresponde al cociente de la media de los $lrd_k(x_i)$ de todos los k vecinos y el **lrd** de ese punto. Es decir,

$$LOF_k(x_i) = \frac{\sum_{x_j \in N_{i,k}} \frac{lrd_k(x_j)}{lrd_k(x_i)}}{N_{i,k}} = \frac{\sum_{x_j \in N_{i,k}} lrd_k(x_j)}{N_{i,k} \cdot lrd_k(x_i)} \tag{8.7}$$

Esto proporciona una medida relativa de la densidad de cada observación en comparación con sus vecinos. Se considera que una observación es anómala si se encuentra en una región de baja densidad (lejos de sus vecinos) mientras que sus vecinos se encuentran en regiones de mayor densidad (con muchos vecinos cerca). Un valor de $LOF_k(x_i)$ mucho mayor que 1 puede ser señal de que la observación tiene un *reachability* promedio mucho mayor en comparación con sus vecinos, por lo que se considera una anomalía.

Una de las ventajas del **LOF** frente a la distancia **kNN** es precisamente la interpretación más sencilla. En este caso tenemos que:

- Si $LOF_k(x_i) > 1$ entonces es probable que la observación i sea una anomalía
- Si $LOF_k(x_i) \leqslant 1$ entonces es poco probable que la observación i sea una anomalía

Así, en la práctica, cuanto más lejos de uno sea el **LOF** para una observación, se considera una anomalía, ya que es significativamente más raro que sus vecinos, indicando que se encuentra en una región de baja densidad. Y entre más cerca esté el

[9] Si hay múltiples observaciones todas exactamente a $d_{i,k}$ de x_i, entonces $N_{i,k}$ puede contener más de k observaciones, pero por simplicidad omitiremos este caso. Es decir, puedes suponer que $N_{i,k} = k$. Por otro lado, nota que una observación x_j puede estar dentro de $N_{i,k}$, mientras que x_i no está en el conjunto $N_{j,k}$ como se puede observar en los puntos verdes del panel c de la Figura 8.3.

[10] Nota que *reachability* es parecida a la distancia euclidiana pero truncada para que no sea menor a $d_{j,k}$.

[11] Nota que el $lrd_k(x_i)$ corresponde a la *reachability* promedio de la observación x_i hacia sus k vecinos más próximos. El denominador de esta expresión es equivalente al denominador representado en el panel d) de la Figura 8.3.

LOF de una observación a 0, se considerará la observación como normal, ya que su densidad local es similar a la de sus vecinos.

Dado que el **LOF** es un método basado en la densidad (local) y no en la distancia como la distancia **kNN**, es más robusto a los **outliers** que la distancia **kNN**. En ambos casos se requiere definir el número de vecinos (k).

8.3 Implementación en R

Para estudiar cómo implementar las aproximaciones estudiadas emplearemos los mismos datos que empleamos en los Capítulos 2, 3 y 4. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalias). La descripción de las variables se encuentra en la Sección 2.4. Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Seleccionemos solo las variables cuantitativas para poder aplicar las técnicas estudiadas en las secciones anteriores. Empleemos la función **select_if()** del paquete *dplyr* (Wickham et al., 2021) para hacer esta tarea más sencilla. Y finalmente estandaricemos los datos empleando la función **scale()** de la base de R.

```
library(dplyr)
german_cuanti_est <- german %>%
    select_if(is.numeric) %>%
    scale()
```

8.3.1 Detectando anomalías con la distancia kNN

Para calcular la distancia **kNN** existen diferentes funciones, en este libro emplearemos la función **kNNdist()** del paquete *dbscan* (Hahsler et al., 2019). Esta función típicamente incluye los siguientes argumentos:

kNNdist(x, k, all)

donde:

- **x**: Una matriz de datos, un objeto de clase **dist** (distancia) o un objeto de clase **kNN**.
- **k**: Número de vecinos.
- **all**: Un valor lógico para expresar si se quieren que se calcule la distancia **kNN** para todos los posibles vecinos hasta el establecido en **k**. Por defecto, **all = FALSE**.

Si como argumento se le entrega a esta función un objeto con datos, entonces la distancia **kNN** será calculada empleando la distancia euclidiana. Si deseamos calcular la distancia **kNN** con otro tipo de distancia, podemos emplear un objeto de clase

dist que previamente haya calculado la distancia con el método deseado. Esta función, a diferencia de la mayoría de las funciones disponibles en otros paquetes, es de las pocas que permite la inclusión de otro tipo de distancias diferentes a la euclidiana.

Para nuestro caso, el código para calcular la distancia **kNN** para 10 vecinos, empleando la distancia euclidiana, es:

```
# Cargar paquete
library(dbscan)

# Calcular la distancia kNN con distancia euclidiana.

anomalias_DkNN <- kNNdist(german_cuanti_est, k = 10)
```

En el objeto `anomalias_DkNN` obtenemos las distancias **kNN** para todas las observaciones. Con la siguiente línea de código podemos ver la observación con la mayor distancia **kNN**:

```
# Identificar el número de la observación con la distancia kNN más grande

which.max(anomalias_DkNN)
```

```
## [1] 757
```

```
# La distancia kNN mas alta

anomalias_DkNN[which.max(anomalias_DkNN)]
```

```
## [1] 3.81735
```

También podemos visualizar las distancias kNN con un *boxplot* (ajustado según Hubert y Vandervieren (2008)) como el que se presenta en la Figura 8.4[12].

En la Figura 8.4 podemos ver que existen 22 valores anómalos. Es decir, observaciones con distancias **kNN** relativamente grandes.

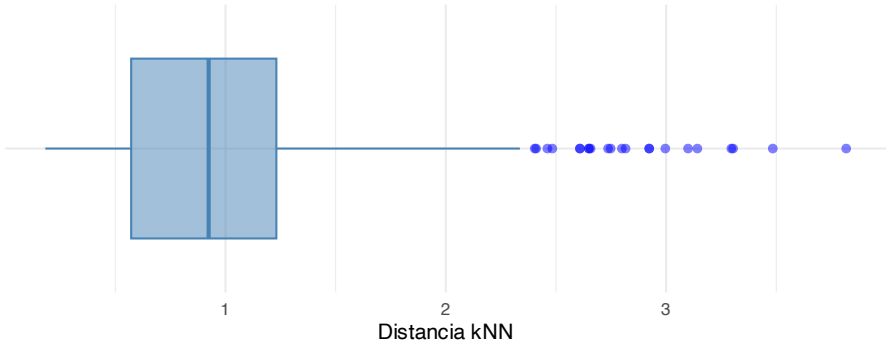
Otra alternativa para visualizar las distancias **kNN** es la función **kNNdistplot()** del paquete *dbscan*; ¡intenta usar esta función!

Nota que el k de 10 fue seleccionado arbitrariamente. En la práctica deberíamos emplear diferentes valores de k y ver qué tan robusta es nuestra decisión. Por ejemplo, calculemos la distancia **kNN** hasta 100 vecinos.

```
# Calcular la distancia kNN con distancia euclidiana hasta 100 vecinos.

anomalias_DkNN_100 <- kNNdist(german_cuanti_est, k = 100, all = TRUE)

# identificar la observación con el mayor kNN para cada uno de los números de
# vecinos (columnas)
```

[12] Intencionalmente se omite el código que genera la Figura 8.4. ¡Intenta reproducir esta visualización!

Figura 8.4. Boxplot ajustado según Hubert y Vandervieren (2008) para la distancia kNN (k=10 y distancia euclidiana)



Fuente: elaboración propia.

```
apply(anomalias_DkNN_100, 2, which.max)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
## 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
## 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
## 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
## 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757 757
```

Podemos observar que sin importar el número de vecinos que empleemos, la observación 757 es considerada una anomalía.

Por otro lado, si queremos emplear otra distancia podemos emplear la función **dist()** que está en el paquete central de R. El argumento **method** determina el tipo de medida de distancia que se quiere emplear. Las opciones para este argumento son: "euclidean", "maximum", "manhattan", "canberra", "binary" o "minkowski". Por defecto, **method = "euclidean"**. Así mismo, está el argumento **p** para establecer la potencia que se desea emplear para la distancia de **Minkowski**, , por defecto **p = 2**.

Por ejemplo, para calcular la distancia **kNN** (para $k = 10$) empleando la distancia de Chebyshev (o Distancia máxima) podemos emplear las siguientes líneas de código:

```
# Calcular Matriz de distancia de Chebyshev
dist_Chebyshev <- dist(german_cuanti_est, method = "maximum")


# Calcular la distancia kNN con distancia de Chebyshev .
```

```
anomalias_DkNN_Chebyshev <- kNNdist(dist_Chebyshev, k = 10)


# Identificar el número de la observación con la distancia kNN más grande

which.max(anomalias_DkNN_Chebyshev)
```

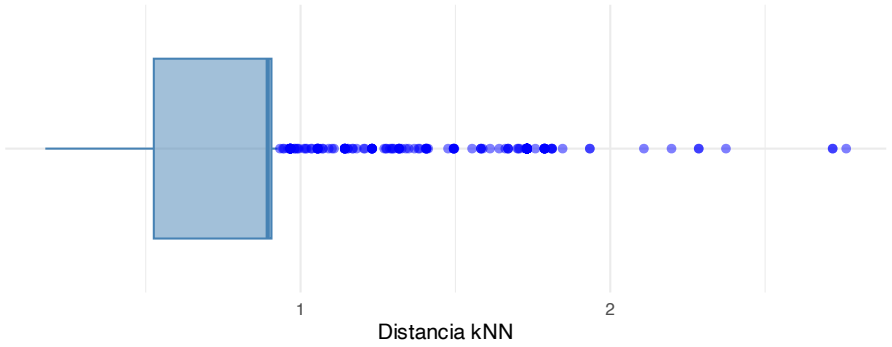
```
## [1] 891
```

```
# La distancia kNN mas alta

anomalias_DkNN_Chebyshev[which.max(anomalias_DkNN_Chebyshev)]
```

```
## [1] 2.761777
```

Y visualicemos estas nuevas distancias kNN con un *boxplot* (ajustado según Hubert y Vandervieren (2008))) como el que se presenta en la Figura 8.5[13].

Figura 8.5. Boxplot ajustado según Hubert y Vandervieren (2008) para la distancia kNN (k=10 y distancia de Chebyshev)



Fuente: elaboración propia.

En la Figura 8.5 podemos ver que existen 185 valores anómalos. Es decir, observaciones con distancias **kNN** relativamente grandes.

8.3.2 Detectando anomalías con LOF

Para calcular **LOF** existen diferentes funciones. No obstante, en este libro, seguiremos empleando las funciones provistas por el paquete *dbscan* (Hahsler et al., 2019). En este caso, la función que nos permite calcular el **LOF** es **lof()**. Esta función típicamente incluye los siguientes argumentos:

lof(x, minPts = 5)

donde:

[13] Intencionalmente se omite el código que genera la Figura 8.4. ¡Intenta reproducir esta visualización!

- **x**: Una matriz de datos, un objeto de clase **dist** (distancia) o un objeto de clase **kNN**.
- **minPts**: Número de vecinos. Al igual que la función **kNNdist()**, si a esta función se le entrega como argumento **x** un objeto con datos, entonces el **LOF** será calculado empleando la distancia euclidiana. Si deseamos calcular el **LOF** con otro tipo de distancia, podemos emplear un objeto de clase **dist** que previamente haya calculado la distancia con el método deseado, como se mostró anteriormente.

En este caso para calcular el **LOF** para nuestros datos con 10 vecinos, tendremos:

```
# Cargar paquete
library(dbscan)

# Calcular LOF con distancia euclidiana.

anomalias_LOF <- lof(german_cuanti_est, minPts = 10)
```

Filtremos las observaciones que tengan un **LOF** mayor a uno.

```
anomalias_LOF_org <- as.data.frame(anomalias_LOF) %>%
    mutate(id = row_number()) %>%
    filter(anomalias_LOF > 1) %>%
    arrange(desc(anomalias_LOF))
```

Nota que en este caso se encuentran 763 observaciones con un **LOF** por encima de 1. La observación con el **LOF** más grande es la 139 con un **LOF** de 4.1198627. Esta observación parece ser una anomalía local. Ahora, puedes chequear tu mismo la consistencia de este resultado para diferentes valores de k. ¡Inténtalo!

8.4 Comentarios finales

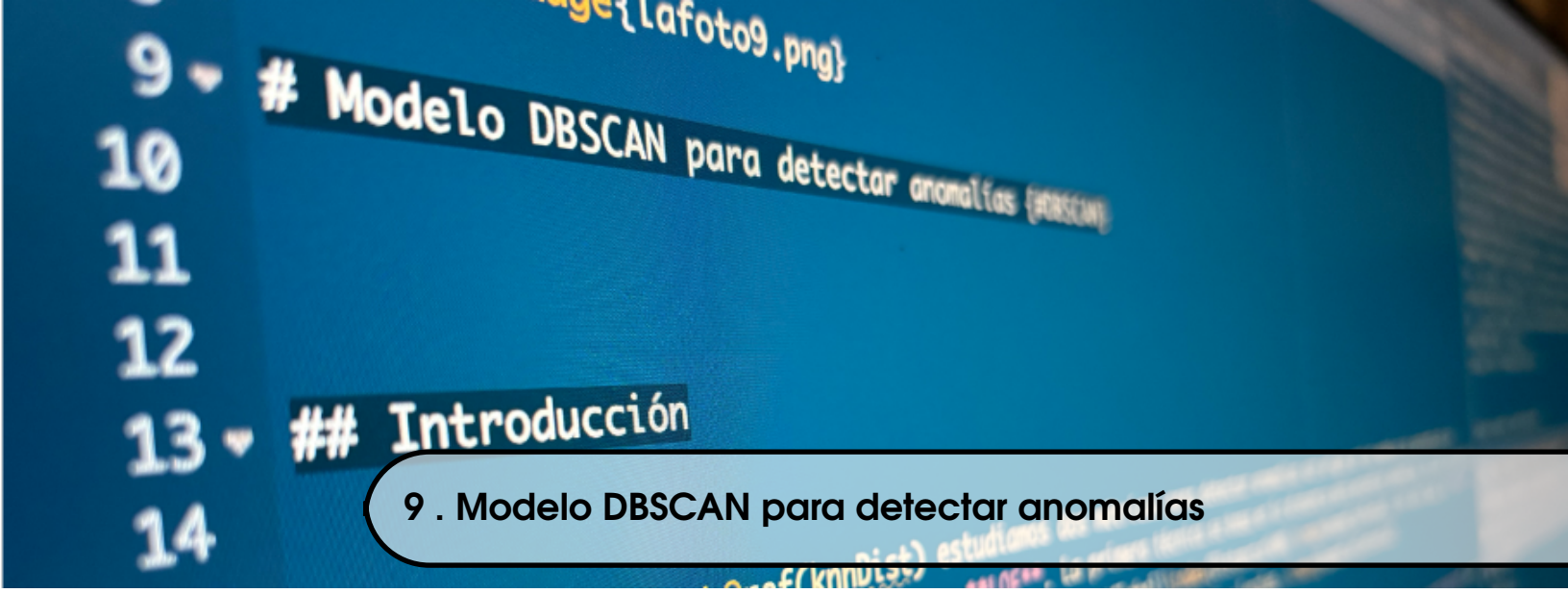
En este capítulo hemos explorado las técnicas de distancia **kNN** y **LOF** para la detección de anomalías multivariadas. Hemos discutido cómo calcular la distancia **kNN** empleando diferentes distancias, como la distancia euclidiana y la de Chebyshev, y cómo representar visualmente las distancias resultantes usando *boxplots* ajustados. La distancia **kNN** permite detectar anomalías globales, así como las técnicas que estudiamos en los Capítulos 2, 3, 4 y 6.

Además, hemos estudiado el proceso de cálculo de **LOF** y la identificación de observaciones anómalas basándonos en un umbral de uno. Esta aproximación permite identificar anomalías locales (también conocidas como contextuales), esta es una gran diferencia con todos los métodos que hemos estudiado hasta el momento en este libro. En los próximos capítulos continuaremos estudiando herramientas que permiten detectar anomalías locales.

El **LOF** es una herramienta poderosa para la detección de anomalías individuales debido a su enfoque en la densidad local, pero esta ventaja es una desventaja para identificar anomalías grupales. Para detectar anomalías grupales, se pueden emplear

técnicas que analicen las relaciones y patrones en subconjuntos más grandes de datos, como lo veremos en los Capítulo 9 y 10.

Ya hemos adquirido una base sólida en la detección univariada y multivariada de anomalías a través de técnicas tanto estadísticas como de aprendizaje de máquina. Sin embargo, vale la pena mencionar que nuestro viaje en este campo está lejos de terminar. En los siguientes capítulos, profundizaremos en estos métodos para ampliar aún más nuestra caja de herramientas para la detección de anomalías.



9 . Modelo DBSCAN para detectar anomalías

9.1 Introducción

En el Capítulo 8 estudiamos dos métricas para detectar anomalías con origen en los modelos de aprendizaje de máquina: la distancia **kNN** y **LOF**. La primera técnica se basa en la distancia para encontrar anomalías, lo cual le permite encontrar anomalías globales. Por otro lado, el **LOF** es un método basado en densidad que permite encontrar anomalías locales.

En este capítulo estudiaremos el uso del modelo **DBSCAN** (*Density-Based Spatial Clustering of Applications with Noise*) para detectar anomalías. El **DBSCAN** es otra técnica basada en densidades que también permite encontrar anomalías locales. Intuitivamente, los métodos basados en densidad identifican regiones de alta concentración de datos y consideran como anomalías aquellos puntos que se encuentran en regiones de baja densidad. De hecho, a diferencia del **LOF**, el método que estudiaremos en este capítulo permite encontrar anomalías grupales. A diferencia del **LOF** que busca la densidad alrededor de cada punto y por eso identifica anomalías locales individuales, esta técnica se concentrará en grupos de observaciones, lo que permitirá encontrar anomalías locales grupales.

El modelo **DBSCAN** es un modelo diseñado para construir clústeres que puede ser empleado en la tarea de detección de anomalías. Es decir, este modelo permite realizar tanto la tarea de clústering[1] como la detección de anomalías.

Inicialmente, el **DBSCAN** fue propuesto por Ester et al. (1996) para identificar clústeres de formas arbitrarias y con robustez ante datos ruidosos. Es decir, datos que no siguen el patrón general o la estructura subyacente de los datos. El ruido en este contexto se refiere a los individuos que no pertenecen claramente a ningún clúster o que pueden interferir con la identificación de clústeres con mayor definición. En otras palabras, una de las características de **DBSCAN** es poder trabajar con datos atípicos que, co-

[1] En Alonso et al. (2025) se presenta una discusión detallada de la tarea de clústering y los diferentes modelos disponibles para realizarla.

mo veremos, el algoritmo termina no clasificando. Una vez que se empezó a usar este algoritmo para hacer clústering, se descubrió que también podía ser empleado para detectar anomalías.

En la Sección 9.2 presentamos el algoritmo **DBSCAN** de manera intuitiva y formal. En la Sección 9.3 presentamos su implementación en R.

9.2 El algoritmo DBSCAN

En general, este tipo de algoritmo basado en densidad se caracteriza por identificar regiones de alta densidad de individuos separadas por regiones de baja densidad en el espacio de datos (Ver Figura 9.1). Al encontrar esas regiones de alta densidad de datos, también encuentra aquellos puntos que no pertenecen a esas zonas de alta densidad (puntos grises en la Figura 9.1). Esas observaciones de baja densidad que no se pueden clasificar en clústeres corresponderán a los datos anómalos localmente; es decir, que no se parecen a sus vecinos.

Así, **DBSCAN**, a diferencia de modelos tradicionales de clústering como los jerárquicos y **k-means**, puede lidiar con los valores anómalos (Alonso et al., 2025). Todos los valores anómalos serán identificados y marcados sin ser clasificados en ningún clúster. Por esta razón, DBSCAN también es empleado para la detección de anomalías (Kassambara, 2017).

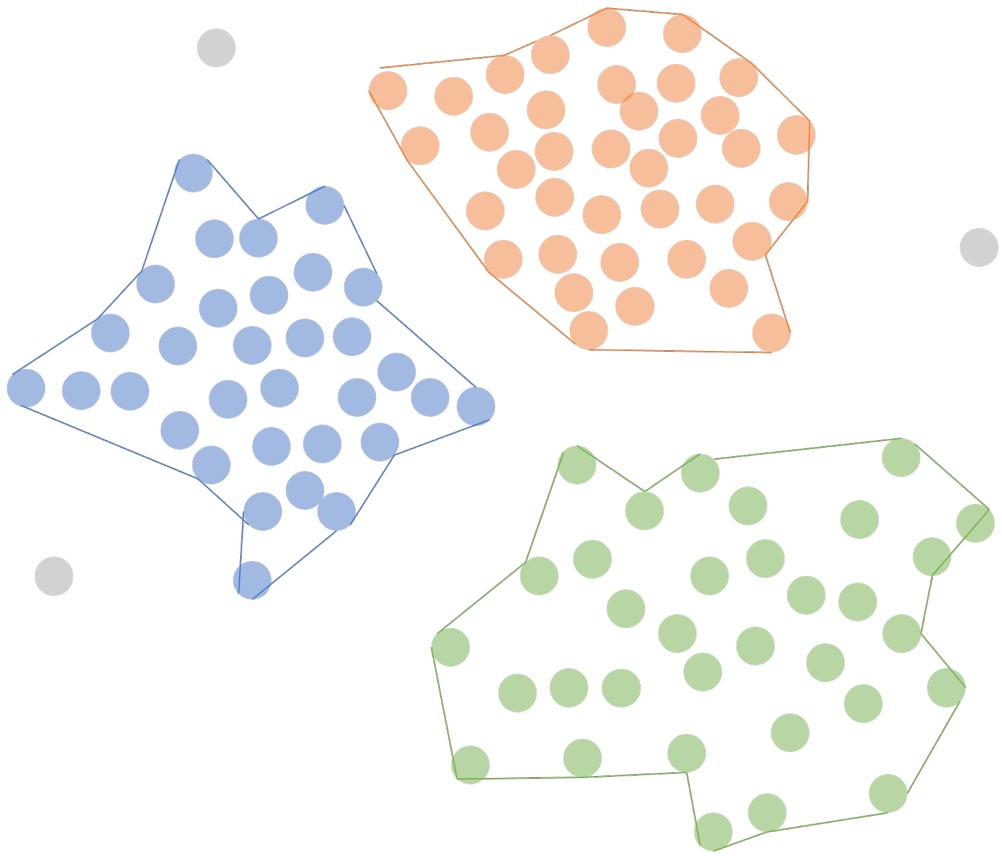
Formalmente, el algoritmo de **DBSCAN** implica dos parámetros importantes: **eps** (ϵ) y **minPts**. **eps** (ϵ) representa el radio máximo del vecindario de una observación. Por otro lado, **minPts** es el número mínimo de puntos que deben estar dentro de la vecindad de un punto para que este sea considerado un punto central y, por tanto, se configure un clúster (incluyendo el mismo punto).

Intuitivamente, cada individuo en la muestra que tenga un número de vecinos (en el vecindario ϵ) igual o mayor a **minPts** será marcado como un punto central (Ver Figura 9.2). Por otro lado, si un individuo w tiene un número de vecinos inferior a **minPts**, pero pertenece al vecindario ϵ (ϵ**-neighborhood**) de otro punto, entonces w será denominado un punto de frontera (*border point*) (Ver Figura 9.2). Y si un punto no es punto central (*core point*) ni uno en la frontera (*border point*), entonces será denominado ruido (*noise point*) o *anomalía* (Ver Figura 9.2). De esta manera, el algoritmo va asignando cada punto a un clúster o lo marca como *anomalía*. Para conocer el detalle de este algoritmo puedes ver el Capítulo 7 de Alonso et al. (2025).

Una de las características del algoritmo **DBSCAN** es que este no requiere la definición previa del número de clústeres, al ser este un resultado del mismo algoritmo. De hecho, el número de clústeres óptimo (q) es resultado del valor óptimo del parámetro **eps** (ϵ). Chen et al. (2020) sugiere emplear la media de las distancias de cada individuo a sus k vecinos más cercanos (distancia **kNN**) para encontrar dicho parámetro.

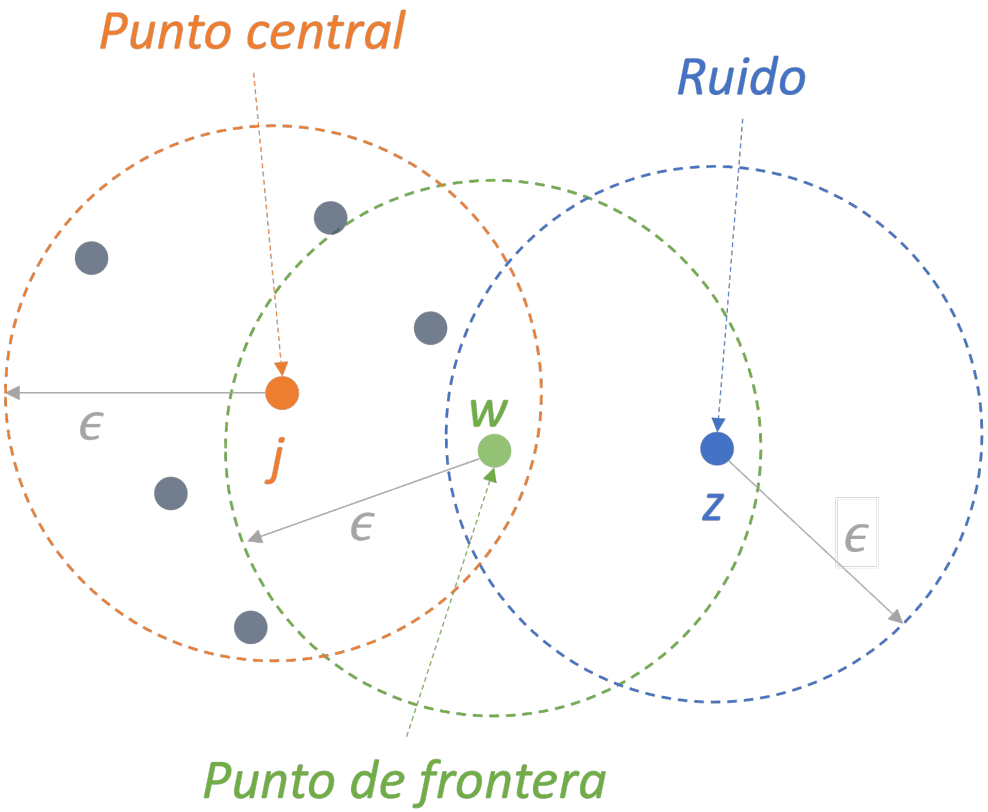
El algoritmo implica calcular la distancia **kNN** para cada individuo y diferentes valores del parámetro **eps** y posteriormente calcular la suma cuadrada de las distancias **kNN**. Empleando ese resultado, se busca el valor de **eps** que minimiza dicha suma.

Figura 9.1. Tipo de clústeres que puede encontrar DBSCAN



Fuente: Alonso, Hoyos y Largo (2024).

Figura 9.2. Clasificación de individuos en el algoritmo DBSCAN con minPts = 6



Fuente: Alonso, Hoyos y Largo (2024).

9.3 Implementación en R

Para detectar anomalías con **DBSCAN** se requiere hacer el ejercicio de clústering, veamos en detalle cómo realizar el procedimiento de clusterización con **DBSCAN** en R. Si ya dominas esta técnica, puedes saltarte esta subsección. Si quieres estudiar en más detalle **DBSCAN** puedes consultar el Capítulo 7 de Alonso et al. (2025).

En este capítulo seguiremos empleando los mismos datos que trabajamos en los Capítulos 2, 3, 4 y 5. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalias). La descripción de las variables se encuentra en la Sección 2.4. Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Seleccionemos solo las variables cuantitativas para poder aplicar **DBSCAN**. Empleemos la función **select_if()** del paquete *dplyr* (Wickham et al., 2021) para hacer esta tarea más sencilla.

```
library(dplyr)
german_cuanti <- german %>%
    select_if(is.numeric)
```

Recuerden que contamos con mil observaciones y 6 variables.

9.3.1 Algoritmo DBSCAN en R

Este algoritmo se puede implementar en R por medio de la función **dbscan()** del paquete *dbscan* (Hahsler et al., 2019). Pero antes de emplear esta función es necesario conocer cuál es el valor óptimo del parámetro **eps**. La propuesta de Chen et al. (2020) se puede implementar en R empleando la función **n_clusters_dbscan** del paquete *parameters*.

Esta función tiene los siguientes argumentos

n_clusters_dbscan(x, standardize, include_factors, eps_range = , distance_method)

donde:

- **x**: objeto con datos de clase **data.frame**
- **standardize**: si es **TRUE** se estandarizarán los datos, este es el valor por defecto.
- **incluir_factores**: si es **TRUE**, las variables de clase **factor** se convierten en objetos numéricos para ser incluidas en los datos para determinar el número de clústeres. Por defecto, es igual a **FALSE**; es decir, se eliminan las variables que sean de clase **factor**. Esto se hace porque la mayoría de los métodos que determinan el número de conglomerados solo funcionan con variables cuantitativas.
- **min_size**: el número mínimo de individuos (incluyendo al mismo individuo) requeridos en el vecindario ϵ (ϵ-**neighborhood**) para ser considerado como un punto central (*core point*). Por defecto, el valor es el 10% de la muestra (**min_size =**

0.1). Si se emplea un entero, este será el número de observaciones. Equivale al parámetro **minPts**.
- **eps_range**: El rango sobre el cual se evaluarán los posibles valores del parámetro **eps**.
- **method**: El método para escoger el parámetro **eps** óptimo. En nuestro caso, empleemos **method = "SS"** para que se calcule la suma cuadrada.
- **distance_method**: El tipo de distancia a calcular. Este elemento es pasado a la función **dist()**. . Por defecto, el método es **"euclidean"**; los otros posibles valores son "maximum", "manhattan", "canberra", "binary" y "minkowski". No obstante, para el caso del algoritmo **DBSCAN** no se encuentra habilitada la opción para otra distancia.

El valor óptimo del parámetro **eps** (ϵ) para el algoritmo **DBSCAN** y el método de la distancia **kNN** (Ver Sección 8.1) y empleando la distancia euclidiana lo podemos encontrar con el siguiente código empleando nuestros datos ya estandarizados (`datos_est`) y la distancia euclidiana.

```
library(parameters)
library(dbscan)
res_dbscan_knn <- n_clusters_dbscan(german_cuanti, standardize = TRUE,
↪  eps_range = c(0.001,
    3), min_size = 5, distance_method = "euclidean", method = "SS")
```

Los resultados del **eps** óptimo los podemos visualizar imprimiendo el resultado en la consola o por medio de la Figura 9.3.

```
res_dbscan_knn
```

```
##           eps n_Clusters total_SS
## 1  0.00100000          0 5994.000
## 2  0.06220408          0 5994.000
## 3  0.12340816          7 5904.835
## 4  0.18461224         10 5614.148
## 5  0.24581633         21 5381.524
## 6  0.30702041         21 5013.082
## 7  0.36822449         24 4717.736
## 8  0.42942857         32 4498.607
## 9  0.49063265         33 4225.445
## 10 0.55183673         32 4011.373
## 11 0.61304082         33 3853.884
## 12 0.67424490         32 3663.358
## 13 0.73544898         33 3511.788
## 14 0.79665306         34 3429.663
## 15 0.85785714         34 3372.487
## 16 0.91906122         10 4475.440
## 17 0.98026531          4 4352.314
## 18 1.04146939          6 4236.618
## 19 1.10267347          6 4167.380
```

```
## 20 1.16387755          7 4146.265
## 21 1.22508163          7 4113.196
## 22 1.28628571          6 4100.781
## 23 1.34748980          5 4078.374
## 24 1.40869388          6 4063.361
## 25 1.46989796          6 4050.378
## 26 1.53110204          6 4035.224
## 27 1.59230612          6 4036.018
## 28 1.65351020          6 4033.557
## 29 1.71471429          6 4036.916
## 30 1.77591837          2 4797.414
## 31 1.83712245          2 4809.183
## 32 1.89832653          2 4816.140
## 33 1.95953061          2 4817.710
## 34 2.02073469          2 4838.101
## 35 2.08193878          2 4846.380
## 36 2.14314286          2 4843.118
## 37 2.20434694          2 4862.773
## 38 2.26555102          2 4878.640
## 39 2.32675510          2 4895.874
## 40 2.38795918          2 4906.858
## 41 2.44916327          2 4906.858
## 42 2.51036735          2 4926.477
## 43 2.57157143          2 4915.415
## 44 2.63277551          2 4938.087
## 45 2.69397959          2 4939.577
## 46 2.75518367          2 4939.577
## 47 2.81638776          1 5963.098
## 48 2.87759184          1 5963.098
## 49 2.93879592          1 5963.098
## 50 3.00000000          1 5963.098
```

```
plot(res_dbscan_knn)
```

```
## NULL
```

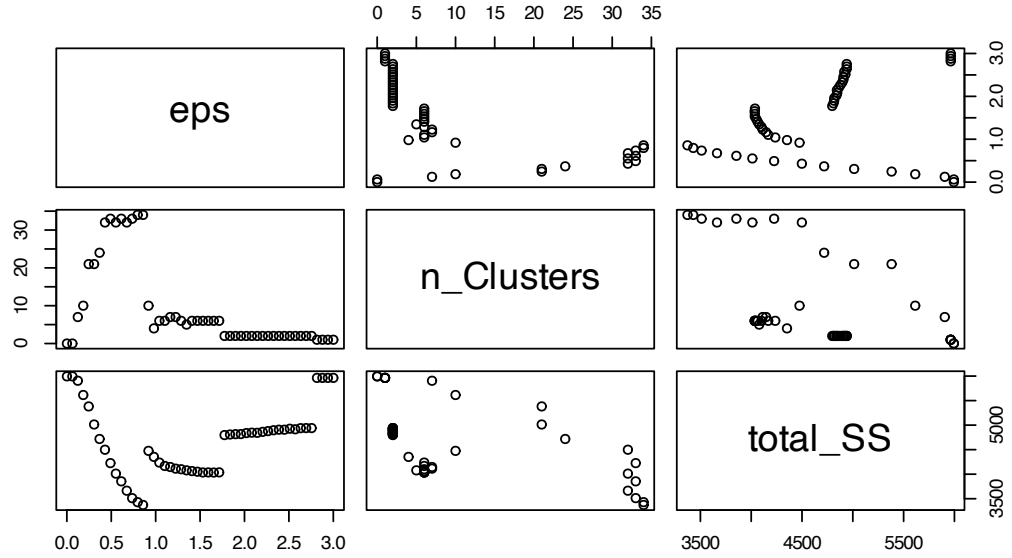
El resultado que minimiza la suma de las distancias **kNN** corresponde a un 0.8578571. Nota que en este caso esto implica 34 clústeres.

El valor óptimo del parámetro **eps** (ϵ) se puede extraer de la siguiente manera:

```
# Extraer el valor óptimo de eps

eps_opt <- attributes(res_dbscan_knn)$eps

eps_opt
```

```
## [1] 0.8578571
```

Figura 9.3. Seleccionando el valor óptimo del parámetro eps por medio de la distancia kNN



Fuente: calculos propios

9.3.2 Detección de anomaías con DBSCAN en R

Ya hemos construido los clústeres, ahora podemos calcular la membrecía de cada individuo. Recuerda que aquellos individuos que no se asignen a un clúster se entenderán como *anomalías*.

La partición de los datos se puede realizar empleando la función **dbscan()** del paquete *dbscan* (Hahsler et al., 2019). La función típicamente incluye los siguientes argumentos:

dbscan(x, eps, minPts)

donde:

- **x**: los datos **estandarizados** o una matriz de proximidad[2].
- **eps**: el parámetro (ϵ) que representa el radio máximo del vecindario.
- **minPts**: el número mínimo de individuos (incluyendo al mismo individuo) requeridos en el vecindario ϵ (ϵ**-neighborhhood**) para ser considerado como un punto central (*core point*).

En nuestro caso, primero debemos estandarizar los datos. Estandaricemos los datos empleando la función **scale()** de la base de R.

```
# Estandarizar los datos
german_cuanti_est <- scale(german_cuanti)
```

[2]Si se quisiera emplear otro tipo de distancia diferente a la euclidiana, esto se puede realizar empleando una matriz de proximidad con la función **dist()**.

Ahora procedamos a particionar las observaciones con el siguiente código:

```
# Instalar el paquete si no se tiene install.packages('dbscan') Cargar el
# paquete

library("dbscan")


# DBSCAN

res_DBSCAN <- dbscan::dbscan(x = german_cuanti_est, eps = eps_opt, minPts = 5)
```

Las membrecías a los clústeres se pueden encontrar fácilmente con el siguiente código:

```
# Resultados de la asignación
head(res_DBSCAN$cluster)
```

```
## [1] 1 2 0 0 0 0
```

```
# Calcular observaciones por clúster
table(res_DBSCAN$cluster)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
## 273  62  32  32  45  43  11   9  71   7  15  99  28  16  37  32  12   6   7   7
##  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  22   6  10   8  15   6  23   6  14   5   6  15   9   6   5
```

Nota que en nuestro caso tenemos 3, 4, 5, 6, 18, 19, 20, 21, 22, 23, 28, 30, 31, 37, 43, 45, 49, 52, 57, 58, 64, 66, 73, 74, 79, 81, 88, 96, 97, 100, 106, 107, 111, 113, 117, 120, 132, 135, 137, 138, 139, 147, 157, 160, 161, 164, 171, 176, 181, 187, 188, 191, 197, 198, 205, 206, 207, 210, 211, 213, 216, 227, 228, 235, 237, 238, 240, 242, 243, 247, 251, 256, 260, 263, 264, 265, 269, 273, 275, 286, 287, 292, 293, 295, 301, 305, 317, 331, 334, 335, 341, 343, 345, 349, 350, 367, 373, 374, 375, 379, 382, 396, 398, 399, 403, 410, 412, 414, 416, 417, 418, 423, 426, 430, 431, 432, 433, 450, 451, 457, 466, 468, 470, 492, 493, 495, 497, 500, 508, 510, 518, 521, 523, 526, 528, 530, 531, 537, 540, 544, 548, 550, 553, 557, 564, 569, 581, 582, 587, 590, 591, 598, 601, 607, 612, 616, 617, 618, 630, 636, 638, 640, 651, 654, 658, 660, 669, 671, 673, 681, 684, 685, 686, 690, 694, 702, 705, 710, 715, 716, 717, 722, 724, 725, 727, 729, 732, 735, 737, 738, 739, 745, 746, 749, 751, 757, 758, 761, 764, 766, 769, 770, 772, 775, 785, 788, 792, 793, 797, 802, 806, 807, 808, 809, 812, 813, 816, 817, 818, 819, 823, 827, 828, 829, 830, 833, 838, 845, 847, 852, 855, 864, 869, 871, 872, 875, 880, 881, 882, 885, 887, 888, 890, 891, 896, 902, 903, 906, 909, 916, 918, 922, 924, 925, 928, 933, 940, 941, 942, 944, 946, 949, 951, 954, 958, 962, 970, 972, 976, 977, 981, 984, 991 observaciones que no han sido clasificadas. Es decir, que fueron asignadas al clúster 0.

Ahora extraigamos el número de los individuos que se pudieron agrupar. Esto lo po-

demos hacer con el siguiente código:

```
# Crear objeto con el índice de las observaciones (número de la observación)
# que son outliers.

DBSCAN_anomal <- as.data.frame(res_DBSCAN$cluster) %>%
    mutate(ID = row_number()) %>%
    rename(Cluster = `res_DBSCAN$cluster`) %>%
    filter(Cluster == 0) %>%
    dplyr::select(ID)

dim(DBSCAN_anomal)
```

```
## [1] 273   1
```

```
head(DBSCAN_anomal, 5)
```

```
##   ID
## 1  3
## 2  4
## 3  5
## 4  6
## 5 18
```

En el objeto `DBSCAN_anomal` tenemos el índice (número) de las 3, 4, 5, 6, 18, 19, 20, 21, 22, 23, 28, 30, 31, 37, 43, 45, 49, 52, 57, 58, 64, 66, 73, 74, 79, 81, 88, 96, 97, 100, 106, 107, 111, 113, 117, 120, 132, 135, 137, 138, 139, 147, 157, 160, 161, 164, 171, 176, 181, 187, 188, 191, 197, 198, 205, 206, 207, 210, 211, 213, 216, 227, 228, 235, 237, 238, 240, 242, 243, 247, 251, 256, 260, 263, 264, 265, 269, 273, 275, 286, 287, 292, 293, 295, 301, 305, 317, 331, 334, 335, 341, 343, 345, 349, 350, 367, 373, 374, 375, 379, 382, 396, 398, 399, 403, 410, 412, 414, 416, 417, 418, 423, 426, 430, 431, 432, 433, 450, 451, 457, 466, 468, 470, 492, 493, 495, 497, 500, 508, 510, 518, 521, 523, 526, 528, 530, 531, 537, 540, 544, 548, 550, 553, 557, 564, 569, 581, 582, 587, 590, 591, 598, 601, 607, 612, 616, 617, 618, 630, 636, 638, 640, 651, 654, 658, 660, 669, 671, 673, 681, 684, 685, 686, 690, 694, 702, 705, 710, 715, 716, 717, 722, 724, 725, 727, 729, 732, 735, 737, 738, 739, 745, 746, 749, 751, 757, 758, 761, 764, 766, 769, 770, 772, 775, 785, 788, 792, 793, 797, 802, 806, 807, 808, 809, 812, 813, 816, 817, 818, 819, 823, 827, 828, 829, 830, 833, 838, 845, 847, 852, 855, 864, 869, 871, 872, 875, 880, 881, 882, 885, 887, 888, 890, 891, 896, 902, 903, 906, 909, 916, 918, 922, 924, 925, 928, 933, 940, 941, 942, 944, 946, 949, 951, 954, 958, 962, 970, 972, 976, 977, 981, 984, 991 observaciones que no se pudieron agrupar. Estas observaciones serán consideradas como *anomalías* grupales locales.

9.4 Comentarios finales

En este capítulo hemos estudiado cómo emplear un método de clústering particionado basado en la densidad de los datos como **DBSCAN** para detectar anomalías locales y multivariadas. Es importante mencionar que los resultados de este algoritmo

son muy sensibles a la escogencia de del parámetro **eps**. Por esta razón, es importante tener una estrategia para fijar, a partir de los datos, este parámetro.

Finalmente, es importante recordar que el algoritmo **DBSCAN** es una herramienta más para hacer la tarea de detectar anomalías que debes tener en tu caja de herramientas. Como las otras herramientas que hemos estudiado, **DBSCAN** no es una solución mágica que funcione en todos los casos, pero tiene algunas ventajas sobre otros algoritmos de agrupamiento.

Por ejemplo, **DBSCAN** no requiere que especifiques el número de vecinos cercanos como si se requiere cuando se emplea la distancia **kNN** y **LOF**. Esto puede ser una gran ventaja, ya que a menudo es difícil saber cuántos vecinos debemos emplear.

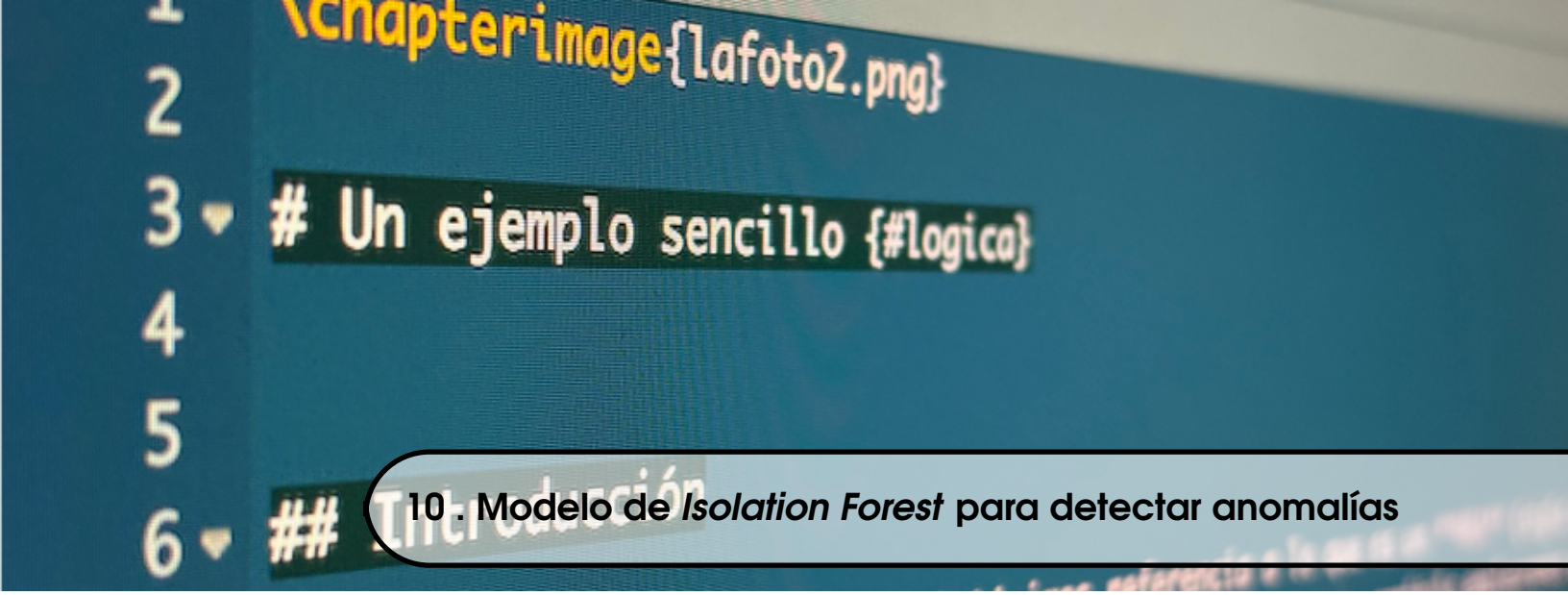
En el siguiente capítulo continuaremos nuestro análisis de algoritmos para detectar anomalías con modelos de aprendizaje de máquina. Estudiaremos una filosofía totalmente diferente.

En este capítulo hemos estudiado cómo emplear un método de clústering particionado basado en la densidad de los datos como **DBSCAN** para detectar anomalías locales y multivariadas. Es importante mencionar que los resultados de este algoritmo son muy sensibles a la escogencia de del parámetro **eps**. Por esta razón, es importante tener una estrategia para fijar, a partir de los datos, este parámetro.

Finalmente, es importante recordar que el algoritmo **DBSCAN** es una herramienta más para hacer la tarea de detectar anomalías que debes tener en tu caja de herramientas. Como las otras herramientas que hemos estudiado, **DBSCAN** no es una solución mágica que funcione en todos los casos, pero tiene algunas ventajas sobre otros algoritmos de agrupamiento.

Por ejemplo, **DBSCAN** no requiere que especifiques el número de vecinos cercanos como si se requiere cuando se emplea la distancia **kNN** y **LOF**. Esto puede ser una gran ventaja, ya que a menudo es difícil saber cuántos vecinos debemos emplear.

En el Capítulo 10 estudiaremos un modelo de ML que sigue una filosofía totalmente diferente: el modelo de **Isolation Forest**.



10. Modelo de *Isolation Forest* para detectar anomalías

10.1 Introducción

En los Capítulos 8 y 9 estudiamos técnicas para detectar anomalías con origen en los modelos de aprendizaje de máquina. La primera técnica de la distancia **kNN** se basa en la distancia para encontrar anomalías, lo cual le permite encontrar anomalías globales. Por otro lado, el **LOF** y el **DBSCAN** son métodos basados en densidad que permiten encontrar anomalías locales.

En este capítulo estudiaremos otra técnica que también es de origen en el aprendizaje de máquina que emplea una filosofía totalmente diferente: **Isolation Forest**. **Isolation Forest** es un método de aprendizaje no supervisado que está inspirado en el algoritmo de clasificación y regresión **Random Forest**[1]. El **Random Forest** es un método de ensamblaje que combina múltiples árboles de decisión para mejorar la precisión y robustez de las predicciones. La idea detrás de un árbol de decisión es dividir los datos en subconjuntos basados en características específicas, creando una estructura de árbol donde cada nodo representa una característica y cada rama representa una decisión basada en esa característica. El objetivo es llegar a una predicción o clasificación en las hojas del árbol. El **Random Forest** crea múltiples árboles de decisión utilizando diferentes subconjuntos de datos y características, y luego combina las predicciones de todos los árboles para obtener una predicción final más precisa.

Al igual que en **Random Forest**, un modelo **Isolation Forest** está formado por la combinación de múltiples árboles llamados **isolation trees** (árboles de aislamiento). La idea principal detrás de **Isolation Forest** es que las anomalías son más fáciles de aislar que las observaciones normales. Por lo tanto, un árbol de aislamiento intenta separar todas las observaciones dividiendo aleatoriamente la región en sectores cada vez más pequeños. La observación que se pueda separar más fácilmente de las demás se considerará como anómala. El principio básico de *Isolation Forest* es que las anomalías suelen requerir menos particiones para ser aisladas que las observaciones normales.

[1] Para una introducción al **Random Forest** puedes ver el Capítulo 8 de Alonso y Hoyos (2025).

En la Sección 10.2 presentamos una introducción a los **Isolation Trees** para después continuar en la Sección 10.3 con el concepto de **Isolation Forest** y posteriormente mostrar cómo se implementa este método en R.

10.2 *Isolation Trees*

Un árbol de aislamiento (**Isolation Tree**) tiene como objetivo aislar las observaciones atípicas, para lo cual de manera recursiva separa las observaciones. La observación que se pueda separar más fácilmente de las demás se considerará como anómala.

Un árbol de aislamiento intenta separar todos los puntos dividiendo aleatoriamente la región (el espacio de los datos) en sectores cada vez más pequeños. El árbol de aislamiento elige una variable (característica o *feature*) y divide las observaciones empleando un valor aleatorio para la característica. Y así continúa la división aleatoria hasta que cada observación se encuentra dentro de su propia subregión, o cada subregión contiene un número máximo de observaciones previamente establecido.

Para entender este algoritmo, veamos un ejemplo. Supongamos que contamos con una muestra de 5 observaciones y dos variables: x_1 y x_2. Además, supongamos que nuestro objetivo es tener al final del proceso solo una observación para cada subregión final (hoja del árbol). En el panel a) de la Figura 10.1 se presenta la muestra con 5 observaciones. Supongamos que el algoritmo escoge aleatoriamente la variable x_1 y divide los datos con un valor aleatorio de 20 (Ver panel b) de la Figura 10.1). Esto divide la muestra en dos. Para valores de x_1 mayores que 20 solo se encuentra una observación (observación en rojo) y para la otra región ($x_1 < 20$) se encuentran 4 observaciones. Como la subregión de la derecha ya alcanzó una observación, esta no se seguirá dividiendo. En otras palabras, la observación en rojo ya fue aislada. Entonces continuamos con la otra región ($x_1 < 20$).

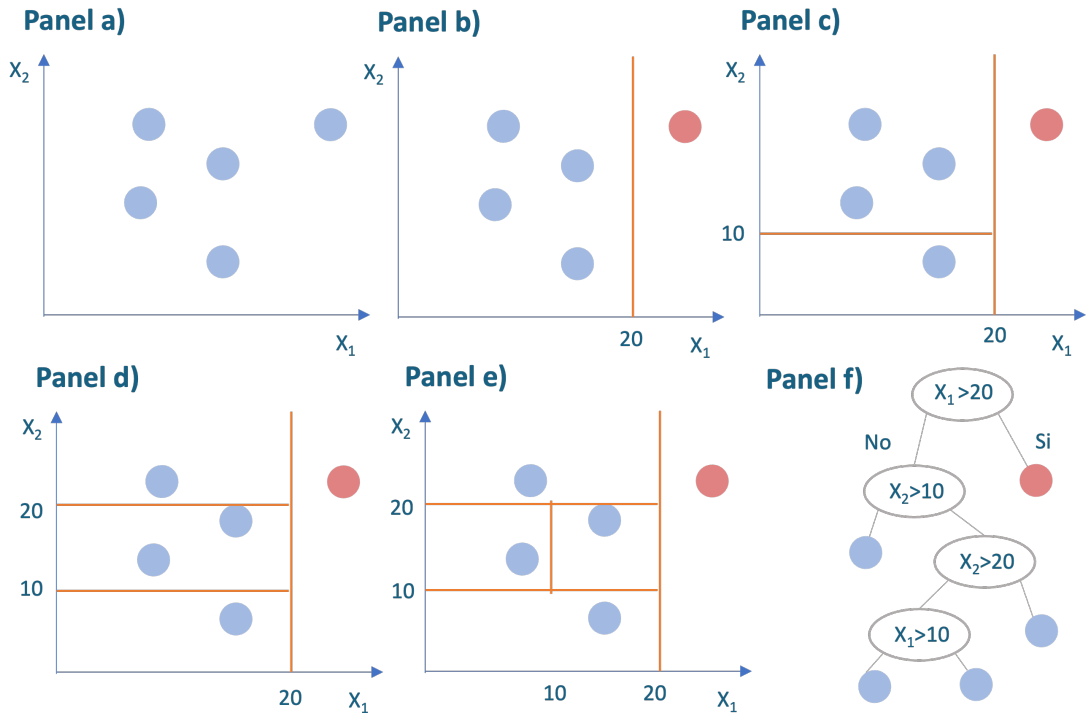
Ahora el algoritmo escogerá nuevamente aleatoriamente una variable y generará un valor aleatorio para partir la subregión. Supongamos que el algoritmo escoge aleatoriamente la variable x_2 y genera un valor aleatorio de 10 (Ver panel c) de la Figura 10.1). El algoritmo continuará dividiendo las subregiones en las que se tengan más de una observación. En nuestro caso, supongamos que aleatoriamente se escoge nuevamente la variable x_2 y genera un valor aleatorio de 20 (Ver panel d) de la Figura 10.1). Finalmente, se escoge la característica x_1 y un valor aleatorio de 10 (Ver panel e) de la Figura 10.1). El algoritmo ha terminado de aislar todas las observaciones. Nota que la observación en rojo, la primera que aislamos, fue la más fácil de aislar y por eso se considerará como un *outlier*.

En el panel f) de la Figura 10.1 se presentan todos los pasos realizados con un diagrama de árbol, de ahí el nombre del algoritmo.

Una vez se ha construido un **Isolation Tree**, se construye un *score* de anomalía para cada observación. Recordemos que un **Isolation Tree** mide el "aislamiento" de una observación en función de la rapidez con la que puede separarse mediante una secuencia de divisiones aleatorias.

El número de divisiones aleatorias necesarias para separar cada observación es un

Figura 10.1. Ejemplo sobre el funcionamiento de un Isolation Tree



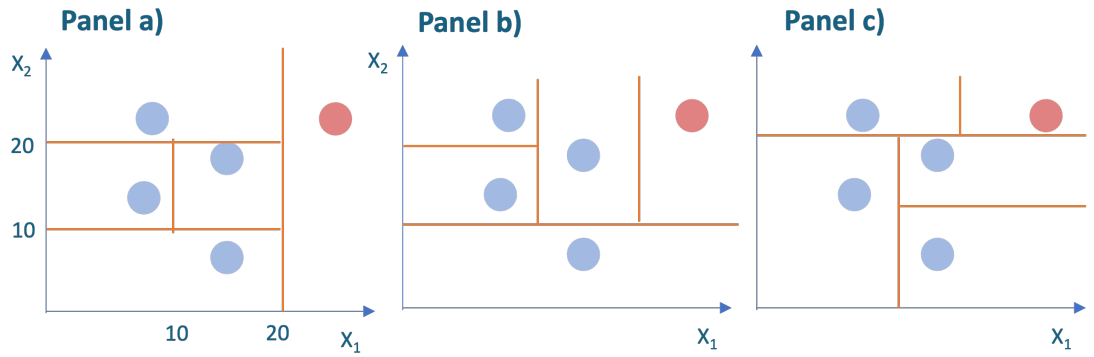
Fuente: elavoración propia.

número entero positivo denominado longitud del camino. La longitud máxima posible del camino depende en parte del tamaño de la muestra empleada para construir el árbol y no tiene una interpretación intuitiva. Por eso, típicamente esas longitudes de los caminos se presentan normalizadas; es decir, entre 0 y 1. Si la puntuación es cercana a 1, entonces la longitud del camino es muy pequeña, lo que significa que el punto se aisló fácilmente con divisiones aleatorias y, por lo tanto, es más probable que sea una anomalía. Si la puntuación es cercana a 0, entonces la longitud del camino es grande, lo que implica que el punto no era anómalo.

10.3 *Isolation Forest*

Regresando al ejemplo que discutimos anteriormente (Ver Figura 10.1). Recordemos que ese aislamiento es fruto de un proceso aleatorio. Por ejemplo, podríamos tener particiones diferentes dependiendo del proceso aleatorio. En la Figura 10.2 se presentan en los paneles b) y c) **Isolation Trees** con particiones diferentes.

Figura 10.2. Ejemplos de diferentes Isolation Trees para la misma muestra



Fuente: elavoración propia.

Como lo discutimos, un **Isolation Tree** "individual" mide el grado de anomalía de una observación en función del número de divisiones aleatorias necesarias para aislarlo. Y como acabamos de ver en la Figura 10.2, las particiones dependen de los valores aleatorios que asigna el algoritmo. Por lo tanto, la puntuación de anomalía de un solo árbol puede ser inestable.

Siguiendo la misma idea de un **Random Forest**, Liu et al. (2008) propusieron emplear muchos árboles de aislamiento en vez de uno solo; es decir, un bosque de árboles de aislamiento (de ahí el nombre de **Isolation Forest**). Un **Isolation Forest** se compone de muchos **Isolation Trees**, cada uno de los cuales "crece" utilizando una muestra diferente de los datos originales.

En este caso, la puntuación de anomalía se genera para cada árbol y luego se calcula la media de todos los árboles cultivados en el bosque. Emplear un bosque para generar muchas puntuaciones y promediarlas tiene dos ventajas. En primer lu-

gar, mientras que los árboles individuales pueden ser sensibles a patrones fortuitos en la muestra de datos, la puntuación media es muy estable. En segundo lugar, nunca se utilizan los datos completos para construir un árbol, por lo que es posible obtener puntuaciones de anomalías mucho más rápidamente. Es decir, un **Isolation Forest** es computacionalmente más eficiente y por tanto puede emplear grandes conjuntos de datos.

Siguiendo la misma idea de un **Random Forest**, Liu et al. (2008) propusieron emplear muchos árboles de aislamiento en vez de uno solo; es decir, un bosque de árboles de aislamiento (de ahí el nombre de **Isolation Forest**). Un **Isolation Forest** se compone de muchos **Isolation Trees**, cada uno de los cuales "crece" utilizando una muestra diferente de los datos originales.

En este caso, la puntuación de anomalía se genera para cada árbol y luego se calcula la media de todos los árboles cultivados en el bosque. Emplear un bosque para generar muchas puntuaciones y promediarlas tiene dos ventajas. En primer lugar, mientras que los árboles individuales pueden ser sensibles a patrones fortuitos en la muestra de datos, la puntuación media es muy estable. En segundo lugar, nunca se utilizan los datos completos para construir un árbol, por lo que es posible obtener puntuaciones de anomalías mucho más rápidamente.

Cuando se entrena un **Isolation Forest** surge la pregunta de cuántos árboles deberíamos "cultivar". Típicamente, el *score* de la anomalía para cada observación tiende a converger después de haber cultivado un número suficientemente grande de árboles. En la práctica, siguiendo a Liu et al. (2008), es común emplear 100 árboles para empezar a investigar si los *scores* están convergiendo o no. En otras palabras, en la práctica se empieza con 100 árboles para determinar si los *scores* ya son estables entre los árboles o no. Y si no es estable se emplean más árboles. Es decir, deberíamos entrenar varios **Isolation Forest** con diferentes números de árboles, y comparar si la puntuación cambia cuando se añaden más árboles. Si la puntuación cambia mucho, esto podría sugerir que se necesitan más árboles. En la siguiente sección mostraremos un ejemplo de cómo hacer esto en R.

10.4 Implementación en R

Para realizar un ejemplo práctico seguiremos usando los mismos datos que empleamos en la mayoría de los Capítulos de este libro. Los datos provienen de Hofmann (1994) y se encuentran en el archivo `datos_credito.RData` que se puede descargar de la página web del libro (https://www.icesi.edu.co/editorial/deteccion-anomalÃŋas). La descripción de las variables se encuentra en la Sección 2.4. Carguemos los datos:

```
load("./datos/datos_credito.RData")
```

Los datos están cargados en el objeto de clase **data.frame** que denominamos `german`. Ese objeto tiene 14 variables y 1000 clientes. Seleccionemos solo las variables cuantitativas para poder aplicar las técnicas estudiadas en las secciones anteriores. Empleemos la función **select_if()** del paquete *dplyr* (Wickham et al., 2021) para hacer esta tarea más sencilla.

```
library(dplyr)
german_cuanti <- german %>%
    select_if(is.numeric)
```

Trabajaremos con una base de datos con 6 variables y 1000 observaciones.

El paquete *isotree* (Cortes, 2024) nos permite entrenar tanto el modelo de **Isolation Tree** como el de **Isolation Forest**.

Para ambos casos, el **Isolation Tree** o el **Isolation Forest**, podemos emplear la función **isolation.forest()** La función típicamente incluye los siguientes argumentos:

isolation.forest(data, ntrees, ndim, seed)

donde:

- **data**: Corresponde a los datos, que pueden ser clase **data.frame**, **matrix**.
- **ntrees**: Corresponde al número de árboles que serán "cultivados". El valor por defecto es **ntrees = 500**
- **ndim**: El número de variables que se emplearán para realizar la separación de las subregiones en cada nodo del árbol (**split**). El valor por defecto es **ndim = 1**.
- **seed**: Semilla que se utilizará para la generación de números aleatorios.

Como se discutió anteriormente, para el número de árboles en un **Isolation Forest** se puede emplear 100 como lo recomiendan Liu et al. (2008). Liu et al. (2010a) sugieren emplear 10 árboles.

De acuerdo a Cortes (2024), el número de árboles ideal debería crecer a medida que hay más variables, se incluyen variables categóricas[2] y el número de variables que se emplearán a la misma vez para separar las subregiones (argumento **ndim** de la función).

Cortes (2024) argumenta que una buena manera de determinar si se ha seleccionado un número adecuado de árboles, es mirar la distribución de los *scores* de anomalía de todas las observaciones. Ellos argumentan, que si el promedio de dichos *scores* es inferior a 0.5 es es señal de que el modelo necesita más árboles.

Por otro lado, el número de variables que se emplearán para realizar la separación de las subregiones en cada nodo del árbol (**split**) es otro parámetro importante para este algoritmo. Si se emplea una variable (el valor por defecto de la función **isolation.forest()**), se entrenará el algoritmo originalmente sugerido por Liu et al. (2008). Liu et al. (2010b) sugieren emplear dos variables en vez de una. Y Hariri et al. (2019) sugieren un número bajo de variables, 2 o 3. Hariri et al. (2019) llamaron a estos modelos con más de una variable en cada *split* como el modelo **Extended Isolation Forest**.

En la práctica, deberemos "jugar" con estos dos parámetros, hasta que encontremos que los resultados converjan y sean robustos a cambios pequeños en ellos.

[2] ¡Sí! este algoritmo puede emplear variables cualitativas. Para comparar los resultados con los otros algoritmos, en este capítulo solo empleamos las variables cuantitativas. Para incluir las variables cualitativas en la función **isolation.forest()** debes emplear el argumento **categ_cols**. Mira la ayuda de esta función para ver somo emplear ese argumento.

Regresando a nuestro ejemplo un **Isolation Tree** puede ser entrenado empleando el siguiente código:

```
library(isotree)
# Entrenar un Isolation Tree
iso_tree <- isolation.forest(german_cuanti, ndim = 1, ntrees = 1, seed = 1234)

iso_tree
```

```
## Isolation Forest model
## Consisting of 1 trees
## Numeric columns: 6
```

Los *scores* para cada observación se pueden calcular empleando la función **predict.isolation_forest** del paquete *isotree* de la siguiente manera:

```
# Calcular scores de anomalía
scores <- predict(iso_tree, german_cuanti)
```

Estos *scores* corresponden a un objeto de clase **numeric** que puede ser analizado de una manera tradicional. Por ejemplo,

```
# Estadísticas descriptivas de los scores
summary(scores)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3664  0.4049  0.4417  0.4758  0.5220  0.8075
```

```
# Observación con el score más grande
which.max(scores)
```

```
## 757
## 757
```

Según este algoritmo, la observación 757 tiene el *score* de anomalía más grande (0.81). Envés de continuar con el análisis de este árbol, calculemos un **Isolation Forest** que sabemos tiene mejores propiedades que un solo árbol.

El siguiente código entrena un **Isolation Forest** con 100 árboles y calcula el *score* de anomalía promedio:

```
# Entrenar un Isolation Forest
iso_forest_100 <- isolation.forest(german_cuanti, ndim = 1, ntrees = 100, seed
↪  = 1234)

iso_forest_100
```

```
## Isolation Forest model
## Consisting of 100 trees
## Numeric columns: 6
```

```
# Calcular scores de anomalía
scores_100 <- predict(iso_forest_100, german_cuanti)
```

En este caso el promedio de los *scores* de anomalía es relativamente cercano a 0.5 (0.48), lo cual según Cortes (2024) es indicio de que el número de árboles es adecuado.

Por otro lado, para seguir la práctica que se discutió arriba, entrenemos un **Isolation Forest** con 500 árboles y comparemos los respectivos *scores* de anomalía para inspeccionar si son parecido o no. Recuerda que antes discutimos que el *score* de anomalías de un **Isolation Forest** no suele cambiar después de que haya crecido un cierto número de árboles. Esto se denomina *convergencia*, y puede comprobarse comparando las puntuaciones generadas por bosques con diferentes números de árboles. Si las puntuaciones difieren mucho, esto podría sugerir que se necesitan más árboles.

Creemos los *scores* de anomalías para bosques con 500 y 1000 árboles con el siguiente código:

```
# Entrenar un Isolation Forest 500 árboles
iso_forest_500 <- isolation.forest(german_cuanti, ndim = 1, ntrees = 500, seed
↪  = 1234)

# Calcular scores de anomalía
scores_500 <- predict(iso_forest_500, german_cuanti)
mean(scores_500)
```

```
## [1] 0.4777459
```

```
# Entrenar un Isolation Forest 1000 árboles
iso_forest_1000 <- isolation.forest(german_cuanti, ndim = 1, ntrees = 1000,
↪  seed = 1234)

# Calcular scores de anomalía
scores_1000 <- predict(iso_forest_1000, german_cuanti)
mean(scores_1000)
```

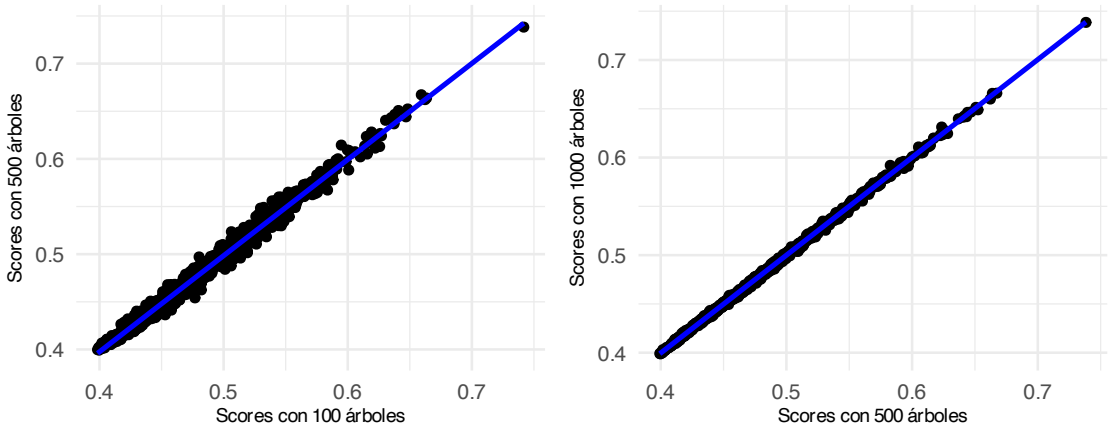
```
## [1] 0.4776192
```

Nota que en todos los tres casos el promedio de los *scores* de anomalía es relativamente cercano a 0.5. En la Figura 10.3 se muestra la comparación de los *scores* de anomalía para los bosques con 100, 500 y 1000 árboles. En la Figura 10.3 podemos ver que los *scores* son algo diferentes entre los bosques de 100 y 500 árboles[3]. No obstante, para los bosques con 500 y 1000 árboles son relativamente parecidos, esto implicaría que el bosque con 500 árboles ya alcanzó la convergencia[4]. En otras palabras, un bosque de 1000 árboles ofrece pocas ventajas sobre uno de 500 árboles.

[3] Los puntos están algo alejados de la línea de 45 grados.

[4] Podríamos seguir chequeando exactamente dónde se alcanza la convergencia, pero realmente eso no es de utilidad para nuestra finalidad de encontrar las anomalías. Lo único importante para cumplir nuestro objetivo es tener claro que el algoritmo ya convergió.

Figura 10.3. Comparación de los scores de anomalía para los modelos de Isolation Forest con 100, 500 y 1000 árboles



Fuente: elavoración propia.

Ahora, podemos emplear el bosque con 500 árboles para realizar nuestro análisis de anomalías. Por ejemplo, las 10 observaciones con el *score* de anomalía más alto se pueden encontrar de la siguiente manera:

```
data.frame(scores_500) %>%
    top_n(10) %>%
    arrange(desc(scores_500))
```

```
##     scores_500
## 757  0.7383126
## 66   0.6673362
## 591  0.6638848
## 654  0.6622151
## 916  0.6524073
## 918  0.6507702
## 928  0.6467596
## 187  0.6441389
## 206  0.6432550
## 891  0.6419446
```

La observación 757 tiene el *score* de anomalía más grande (0.74). Esta observación debería estudiarse en más detalle.

10.5 Comentarios finales

En este capítulo hemos estudiado otro algoritmo para detectar anomalías que tiene su origen en el aprendizaje de máquina, el **Isolation Forest**. Este algoritmo puede identificar anomalías univariadas y multivariadas. También permite identificar anoma-

lías puntuales, anomalías locales o contextuales y anomalías colectivas[5].

Este modelo, más que un modelo, es una familia de modelos en la que los investigadores han desarrollado recientemente grandes contribuciones. Existen diferentes variantes del **Isolation Forest**, conocido en la literatura también como **iForest**. Ya hemos discutido el **Extended Isolation Forest** (**EIF**) que implica emplear dos variables en cada *split* de los datos. El **EIF** lo podemos implementar con la función **isolation.forest()** empleando el argumento **ndim=2** y los argumentos **coefs="uniform"** y **standardize_data=False**. En este caso debemos estar seguros nuevamente de que el algoritmo converge, y no necesariamente convergerá con el mismo número de árboles que la versión anterior. ¡Inténtalo!

Existen otras diferentes versiones del modelos **iForest** que se pueden implementar en con el paquete *isotree*:

- **Split-Criterion iForest** (**SCiForest**): Este modelo realiza las divisiones de las regiones con respecto a un hiperplano elegido al azar en lugar de hacer solo divisiones paralelas al eje. Este modelo propuesto por Liu et al. (2010c) se puede implementar con la función **isolation.forest()** empleando los argumentos: **ndim=2**, **coefs="normal"**, **ntry=10**, **penalize_range=True** y **prob_pick_avg_gain=1**.
- **Fair-Cut Forest** (**FCF**): Este modelo propuesto por Cortes (2021) elige el punto de división según criterios relacionados con las desviaciones estándar o la densidad. Este modelo propuesto por Liu et al. (2010c) se puede implementar con la función **isolation.forest()** empleando los argumentos: **ndim=1**, **coefs="normal"**, **ntry=1** y **prob_pick_pooled_gain=1**.
- **Robust Random-Cut Forest** (**RRCF**): Guha et al. (2016) propusieron este método que implica elegir la variable a dividir con una probabilidad proporcional al rango que abarca en lugar de elegirla uniformemente al azar, siguiendo la idea de que las variables con un rango más amplio son más importantes en una distribución multivariante y más propensas a diferenciar valores atípicos. Este modelo propuesto se puede implementar con la función **isolation.forest()** empleando los argumentos: **ndim=1** y **prob_pick_col_by_range=1**.

En todos los casos, al implementar estos algoritmos será importante chequear que el algoritmo tenga el número de árboles adecuados; es decir, que converja.

Estos modelos basados en bosques de **Isolation Trees** pueden adaptarse a los cambios en la distribución de los datos, por lo que son adecuados para supervisar entornos dinámicos. También son algoritmos flexibles al permitir el uso de variables cualitativas. Adicionalmente, estos bosques son robustos al ruido, lo que puede ser especialmente útil para detectar anomalías en flujos de datos en tiempo real.

Por otro lado, uno de los problemas de esta familia de algoritmos es que son modelos de "caja negra". Es decir, comprender el razonamiento que subyace a su detección de anomalías es poco intuitivo. Esta falta de interpretación puede hacer que no sean

[5]Si bien el algoritmo **Isolation Forest** ha sido construido para detectar anomalías puntuales y contextuales, también puede detectar anomalías colectivas si las características del grupo hacen que sean fáciles de aislar.

útiles en industrias donde la transparencia de los algoritmos y la capacidad de explicarlos sean necesarios, como por ejemplo en los bancos y el sistema financiero o el sector salud.

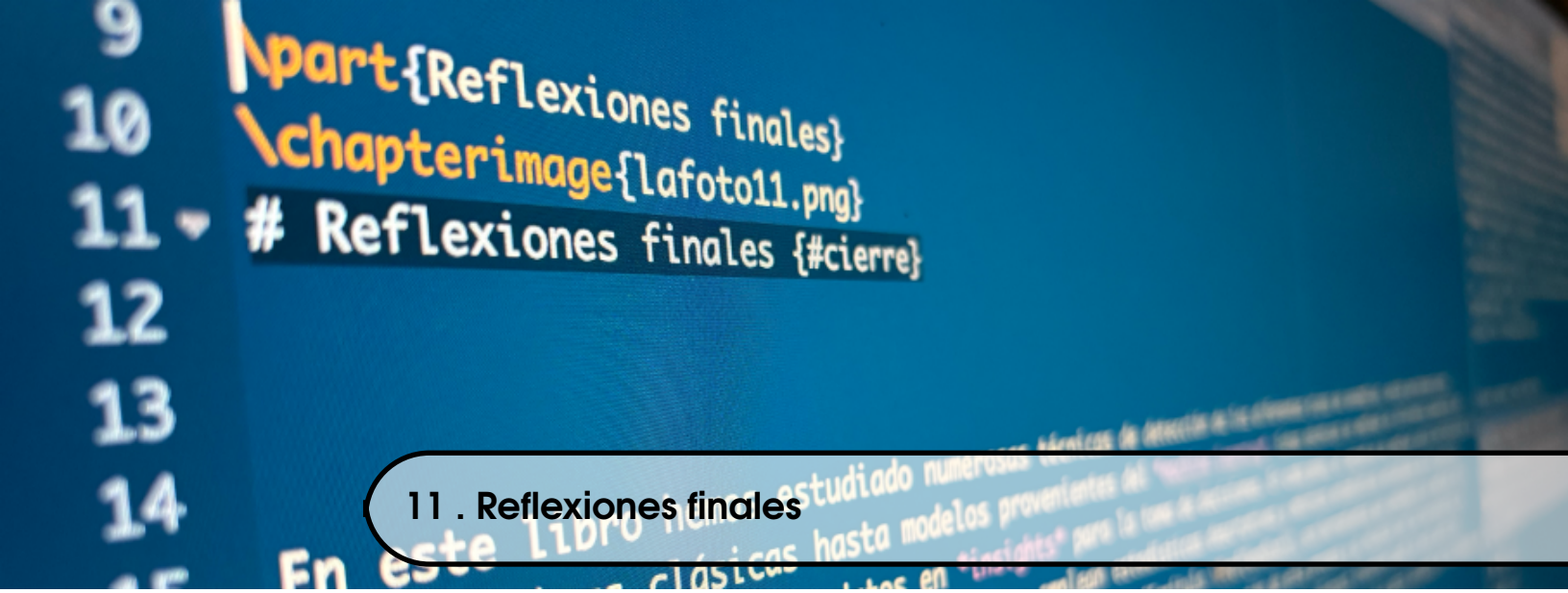
Otro posible problema de los bosques de **Isolation Trees** puede aparecer cuando contamos con muchas variables (alta dimensionalidad de los datos). En estos casos las demandas de capacidad de cómputo de estos modelos serán muy altas y en ocasiones prohibitivas.

El algoritmo de *Isolation Forest* representa un avance significativo en la detección de anomalías de alta dimensión (muchas variables). Su eficiencia y escalabilidad lo convierten en una alternativa atractiva frente a métodos basados en distancias o densidades. Sin embargo, como todo modelo, requiere una validación cuidadosa y debe complementarse con otras técnicas y con el conocimiento del negocio. Emplear al mismo tiempo diferentes técnicas de detección de anomalías permite tener una mirada desde diferentes ángulos al problema de detección de anomalías.

En el Capítulo 11 presentaremos unas reflexiones sobre todas las herramientas que hemos estudiado y sobre las perspectivas de esta amplia rama del *business analytics*. Por ahora recuerda que todas las herramientas estudiadas hasta el momento son útiles para detectar anomalías, cada una tiene unas fortalezas y debilidades frente a las otras. Por eso, para responder una pregunta de negocios que implique la tarea de detección de anomalías siempre será importante contar con una caja de herramientas bien dotada de diferentes aproximaciones para encontrar las anomalías. Dependiendo de los datos y del contexto del negocio, existirán algunas herramientas que tengan más sentido que otras. En últimas, no existe una herramienta para la detección de anomalías que sea siempre mejor que las demás.

Parte IV

Reflexiones finales



11 . Reflexiones finales

En este libro hemos estudiado numerosas técnicas de detección de los diferentes tipos de anomalías, desde aproximaciones estadísticas clásicas hasta modelos provenientes del *machine learning*. Estas técnicas se emplean en diferentes momentos del proceso de transformar datos en *insights* para la toma de decisiones. En cada paso, el objetivo de emplear estas herramientas es diferente. Las técnicas estudiadas que emplean estadísticas descriptivas y métricas estadísticas estudiadas en el Capítulo 2, así como la distancia **kNN** (Capítulo 8), son herramientas que comúnmente se emplean en el proceso de **ETL**[1] (Ver Figura 11.1). En esta parte del proceso, la detección de anomalías tiene como fin garantizar la consistencia e integridad de los datos cargados en las Bodegas de datos.

La herramientas para la detección de anomalías también son empleadas en el **EDA**[2] con el fin de de encontrar datos atípicos que podrían afectar los resultados del análisis exploratorio y los modelos que posteriormente se entrenen o estimen. En el **EDA** típicamente se emplean herramientas como las métricas y estadísticas descriptivas estudiadas en el Capítulo 2, las pruebas estadísticas (Capítulo 3), las técnicas estadísticas para detectar **outliers** multivariados (Capítulo 4), la técnica del **PCA** (Capítulo 5) y la distancia **kNN** y el **LOF** (Capítulo 8). En la parte inferior derecha de la Figura 11.1 se muestra una representación de esto.

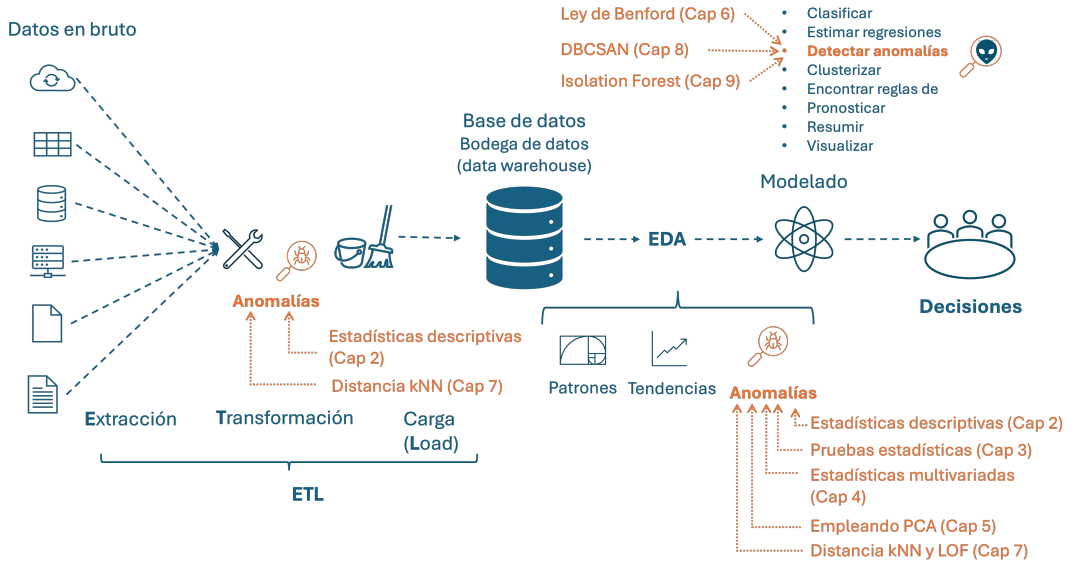
Finalmente, las herramientas de detección de anomalías son empleadas en el modelado como tal de los datos, que es la fase en la que nos concentramos en responder a la pregunta de negocio. En esta fase típicamente se emplean modelos como la Ley de Benford (Capítulo 6), el modelo **DBSCAN** (Capítulo 9) y el **Isolation Forest** (Capítulo 10) (Ver parte superior derecha de la Figura 11.1).

De esta manera hemos recorrido un largo camino para estudiar técnicas de detec-

[1] Ver la Sección 3.1 para una discusión de este proceso y su relación con las herramientas de detección de anomalías.

[2] En la Sección 2.1 se presenta una descripción de este proceso y su relación con las herramientas de detección de anomalías.

Figura 11.1. La detección de anomalías en el business analytics



Fuente: elaboración propia.

ción de anomalías que son útiles para los científicos de datos no solo para responder preguntas de negocio relacionadas con la tarea de detección de anomalías, sino en su trabajo cotidiano que implique el **EDA** y eventualmente el proceso de **ETL**.

Si bien el libro ha cubierto una parte fundamental de los modelos disponibles para detectar anomalías, es importante reconocer que existen otros caminos no explorados aquí. Por un lado, dejamos fuera a los modelos supervisados de detección de anomalías, donde el algoritmo aprende a partir de ejemplos etiquetados de observaciones "normales" y "anómalas". Estos métodos, que incluyen desde regresiones logísticas hasta *Random Forests*, constituyen un campo vasto que emplea los modelos de clasificación. Estos modelos son desarrollados en detalle en Alonso y Hoyos (2025). Te invitamos a estudiar este texto para complementar tu formación en la tarea de detección de anomalías.

Por otro lado, incluso dentro del aprendizaje no supervisado y de la estadística, existen múltiples aproximaciones que no han sido discutidas. Desde métodos robustos basados en teoría de valores extremos y funciones de densidad, hasta técnicas más recientes de *autoencoders*[3] y *variational autoencoders*[4]. La riqueza de modelos posibles

[3]Un *autoencoder* es una red neuronal diseñada para aprender una representación comprimida de los datos (codificación) y luego reconstruirlos (decodificación). La idea es que, al obligar al modelo a comprimir y luego reconstruir, se revelen patrones ocultos y se identifiquen desviaciones: las anomalías suelen reconstruirse mal, lo que permite detectarlas.

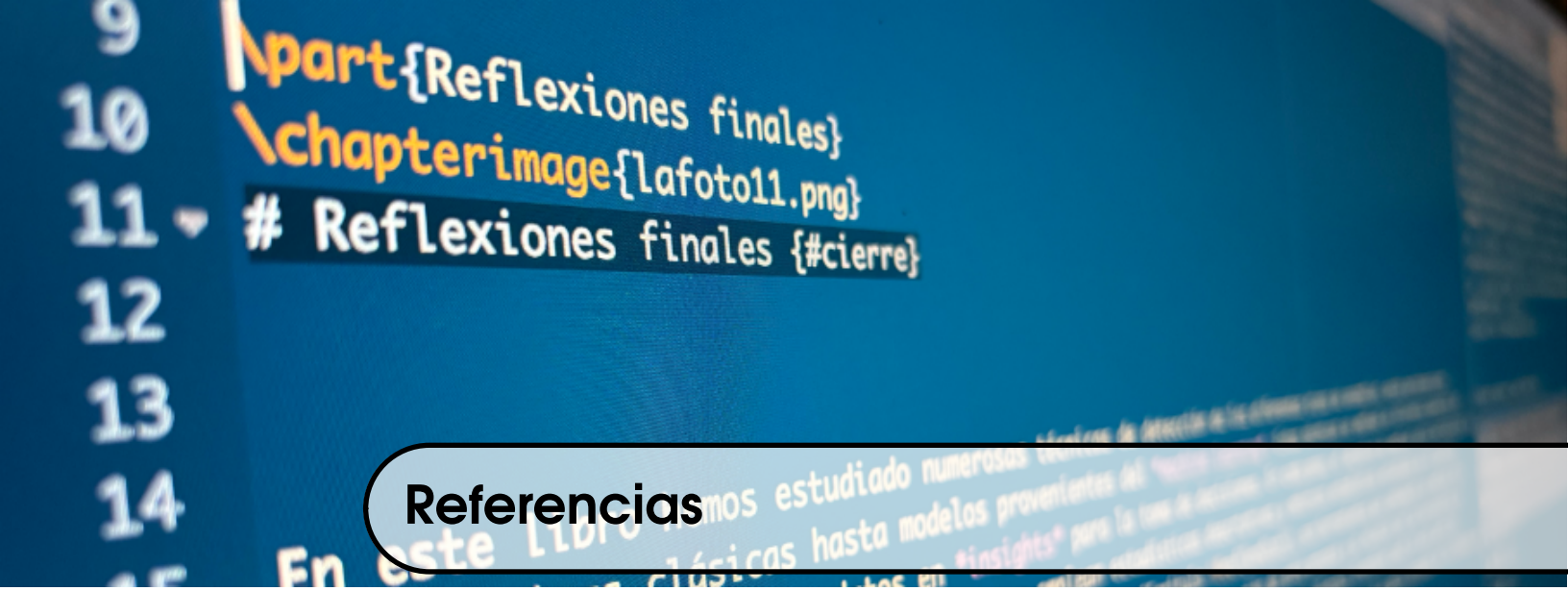
[4]Las *redes neuronales variacionales* (en inglés, *variational autoencoders*) son un tipo de red neuronal probabilística que no solo comprime y reconstruye los datos, sino que además aprende la distribución estadística subyacente. Esto permite generar nuevas observaciones similares a las originales y, en el contexto de

refleja que la detección de anomalías sigue siendo un territorio de innovación permanente.

El horizonte de este campo se amplía aún más con la irrupción de los LLM, como GPT o Gemini (Ver Capítulo 7 para una discusión de estos modelos). Aunque su uso en detección de anomalías todavía está en construcción, se vislumbran aplicaciones en la generación de características (*feature engineering*) asistida, la explicación de patrones complejos y la integración de datos heterogéneos (como por ejemplo textos, imágenes y series de tiempo) en modelos híbridos. Es probable que en los próximos años veamos algoritmos que combinen la potencia predictiva de los LLM con la precisión de los métodos estadísticos y de *machine learning*.

Con esta obra esperamos haber fortalecido tu caja de herramientas como científico de datos. La detección de anomalías, al final, es una forma de encontrar "la aguja en el pajar", y dominar diferentes enfoques te permitirá enfrentar problemas diversos, desde la limpieza de datos hasta la identificación de fraudes o fallas críticas en sistemas. Y recuerda, en el mundo del *business analytics*, ¡la imaginación es el límite!

anomalías, identificar aquellas que se desvían de la distribución aprendida.



Referencias

Almeida, A., Loy, A., y Hofmann, H. (2018). *ggplot2 Compatible Quantile-Quantile Plots in R*.

Alonso, J. C. (2020). Herramientas del Business Analitycs en R: Análisis de Componentes Principales para resumir variables. Icesi Economics Lecture Notes 18188, Universidad Icesi.

Alonso, J. C. (2022). *Empezando a transformar bases de datos con R y dplyr*. Universidad Icesi.

Alonso, J. C. (2024). *Introducción al Modelo Clásico de Regresión para Científico de Datos en R*. Universidad Icesi.

Alonso, J. C. y Arboleda, A. M. (2025). *Introducción al Análisis de Canastas de Compra para analytics translators y científicos de datos (empleando R)*. Universidad Icesi.

Alonso, J. C. y Hoyos, C. C. (2025). *Una introducción a los modelos de clasificación empleando R*. Universidad Icesi.

Alonso, J. C., Hoyos, C. C., y Largo, M. F. (2025). *Una introducción a los modelos de Clústering empleando R*. Universidad Icesi.

Alonso, J. C. y Largo, M. F. (2023). *Empezando a visualizar datos con R y ggplot2*. Universidad Icesi, 2. edition.

Alonso, J. C. y Ocampo, M. P. (2022). *Empezando a usaR: Una guía paso a paso*. Universidad Icesi.

Alonso, J. C. y Serrano, E. (2025). *Del dato a la decisión: Business Analytics para Analytics Translators*. Universidad Icesi.

Benford, F. (1938). The law of anomalous numbers. *Proceedings of the American philosophical society*, pages 551–572.

Benford, F. (2012). The Law of Anomalous Numbers. *Proceedings of the American Philosophical Society*, 78(4):551–572.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., y Liang, P. (2022). On the opportunities and risks of foundation models.

Bougeard, S. y Dray, S. (2018). Supervised multiblock analysis in R with the ade4 package. *Journal of Statistical Software*, 86(1):1–17.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., y Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA. Association for Computing Machinery.

Cabana, E., Lillo, R. E., y Laniado, H. (2021). Multivariate outlier detection based on a robust mahalanobis distance with shrinkage estimators. *Statistical papers*, 62:1583–1609.

Casas, P. (2024). *funModeling: Exploratory Data Analysis and Data Preparation Tool-Box*. R package version 1.9.5.

Chalmers, R. P. y Flora, D. B. (2015). faoutlier: An r package for detecting influential cases in exploratory and confirmatory factor analysis. *Applied Psychological Measurement*, 39(7):573–574.

Chen, Y., Ruys, W., y Biros, G. (2020). Knn-dbscan: a dbscan in high dimensions. *arXiv preprint arXiv:2009.04552*.

Cinelli, C. (2018). *benford.analysis: Benford Analysis for Data Validation and Forensic Analytics*. R package version 0.1.5.

Cortes, D. (2021). Revisiting randomized choices in isolation forests.

Cortes, D. (2024). *isotree: Isolation-Based Outlier Detection*. R package version 0.6.1-1.

Darrouzet-Nardi, A. (2018). *hotspots: Hot Spots*. R package version 1.0.3.

Dinno, A. (2018). *paran: Horn's Test of Principal Components/Factors*. R package version 1.5.2.

Dixon, W. J. (1950). Analysis of extreme values. *The Annals of Mathematical Statistics*, 21(4):488–506.

Dixon, W. J. (1951). Ratios involving extreme values. *The Annals of Mathematical Statistics*, 22(1):68–78.

D'Orazio, M. (2022). *univOutl: Detection of Univariate Outliers*. R package version 0.4.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

EStimator, D. (1999). A fast algorithm for the minimum covariance. *Technometrics*, 41(3):212.

Filzmoser, P. (2023). *chemometrics: Multivariate Statistical Analysis in Chemometrics*. R package version 1.4.4.

Fox, J. y Weisberg, S. (2019). *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, third edition.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21.

Guha, S., Mishra, N., Roy, G., y Schrijvers, O. (2016). Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pages 2712–2721. PMLR.

Hahsler, M., Piekenbrock, M., y Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91(1):1–30.

Hampel, F. R. (1974). The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.

Hariri, S., Kind, M. C., y Brunner, R. J. (2019). Extended isolation forest. *IEEE transactions on knowledge and data engineering*, 33(4):1479–1489.

Harrell, F. E. y Davis, C. E. (1982). A new distribution-free quantile estimator. *Biometrika*, 69(3):635–640.

Hill, T. P. (1995). A Statistical Derivation of the Significant-Digit Law. *Statistical Science*, 10(4):354–363.

Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2):179–185.

HSN-Consultants (2021). Nilson report: Card fraud worldwide. Technical report, HSN Consultants, Inc.

Hubert, M. y Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational statistics & data analysis*, 52(12):5186–5201.

Joenssen, D. W. (2015). *BenfordTests: Statistical Tests for Evaluating Conformity to Benford's Law*. R package version 1.2.0.

Kassambara, A. (2017). *Practical guide to cluster analysis in R: Unsupervised machine learning*, volume 1. Sthda.

Kassambara, A. y Mundt, F. (2020). *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.7.

Komsta, L. (2022). *outliers: Tests for Outliers*. R package version 0.15.

Kossovsky, A. E. (2021). On the Mistaken Use of the Chi-Square Test in Benford's Law. *Stats*, 4(2):419–453.

Lê, S., Josse, J., y Husson, F. (2008). FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18.

Leys, C., Klein, O., Dominicy, Y., y Ley, C. (2018). Detecting multivariate outliers: Use a robust variant of the mahalanobis distance. *Journal of Experimental Social Psychology*, 74:150–156.

Lin, W. (2024). *mt: Metabolomics Data Analysis Toolbox*. R package version 2.0-1.20.

Liu, F., Ting, K., y Zhou, Z. (2010a). Can isolation-based anomaly detectors handle arbitrary multi-modal patterns in data. *Technical Report*.

Liu, F. T., Ting, K. M., y Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.

Liu, F. T., Ting, K. M., y Zhou, Z.-H. (2010b). On detecting clustered anomalies using sciforest. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II 21*, pages 274–290. Springer.

Liu, F. T., Ting, K. M., y Zhou, Z.-H. (2010c). On detecting clustered anomalies using sciforest. In Balcázar, J. L., Bonchi, F., Gionis, A., y Sebag, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 274–290, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lucas, A. (2022). *amap: Another Multidimensional Analysis Package*. R package version 0.8-19.

Maechler, M., Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., Conceicao, E. L. T., y Anna di Palma, M. (2024). *robustbase: Basic Robust Statistics*. R package version 0.99-2.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences* (*Calcutta*), 2:49–55.

Mclachlan, G. (1999). Mahalanobis distance. *Resonance*, 4:20–26.

Millard, S. P. (2013). *EnvStats: An R Package for Environmental Statistics*. Springer, New York.

Mukhamediev, R. I., Popova, Y., Kuchin, Y., Zaitseva, E., Kalimoldayev, A., Symagulov, A., Levashenko, V., Abdoldina, F., Gopejenko, V., Yakunin, K., et al. (2022). Review of artificial intelligence and machine learning technologies: Classification, restrictions, opportunities and challenges. *Mathematics*, 10(15):2552.

Nenadic, O. y Greenacre, M. (2007). Correspondence analysis in r, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, 20(3):1–13.

Newcomb, S. (1881). Note on the Frequency of Use of the Different Digits in Natural Numbers. *American Journal of Mathematics*, 4(1):39–40.

Nigrini, M. J. (2012). *Benford's Law: Applications for forensic accounting, auditing, and fraud detection*, volume 586. John Wiley & Sons.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ramaswamy, S., Rastogi, R., y Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438.

Rorabacher, D. B. (1991). Statistical treatment for rejection of deviant values: critical values of dixon's"q"parameter and related subrange ratios at the 95% confidence level. *Analytical chemistry*, 63(2):139–146.

Rosner, B. (1975). On the detection of many outliers. *Technometrics*, 17(2):221–227.

Rousseeuw, P. (1984). Least median of squares regression. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 79:871–880.

Rousseeuw, P. y Zomeren, B. (1990). Unmasking multivariate outliers and leverage points. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 85:633–639.

Rousseeuw, P. J. y Van Zomeren, B. C. (1990). Unmasking multivariate outliers and leverage points. *Journal of the American Statistical association*, 85(411):633–639.

Tietjen, G. L. y Moore, R. H. (1972). Some grubbs-type statistics for the detection of several outliers. *Technometrics*, 14(3):583–597.

Tukey, J. W. (1970). *Exploratory data analysis - preliminary edition*. Addison-Wesley.

Tukey, J. W. (1977). *Exploratory data analysis*, volume 2. Addison-Wesley Series in Behavioral Science: Quantitative Methods.

Venables, W. N. y Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H., François, R., Henry, L., y Müller, K. (2021). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.7.

William Revelle (2023). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package version 2.3.9.

Yutani, H. (2022). *gghighlight: Highlight Lines and Points in 'ggplot2'*. R package version 0.4.0.

Índice alfabético

Universidad Icesi

Editorial Universidad Icesi

La tarea de detección de anomalías tiene como finalidad encontrar al individuo (observación) que se comporta de manera diferente a los demás. En otras palabras, se emplea para encontrar aquel caso que no sigue el patrón de las otras observaciones. En el mundo del *business analytics* nos enfrentamos a preguntas de negocios que implican identificar lo inesperado, lo excepcional, lo que se desvía de lo habitual. Este desafío es fundamental en muchas áreas de las organizaciones: por ejemplo, en la planta de producción se desea detectar el producto anómalo y en el área contable se desea encontrar fraudes. La identificación de lo inesperado, lo excepcional, lo que se desvía de lo habitual, se conoce como la tarea de detección de anomalías.

Herramientas del BIG DATA y analytics