Introducción a la ciencia de datos en R. Un enfoque práctico

Introducción a la ciencia de datos en R. Un enfoque práctico

José Nelson Pérez Castillo

ESPACIOS

Dedicatoria

A mis estudiantes de la Universidad Distrital Francisco José de Caldas, fuente de inspiración para aproximarme a una metodología de investigación reproducible

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

UD Editorial

ESPACIOS

Primera edición, mayo de 2020
ISBN: 978-958-787-171-5

Dirección Sección de Publicaciones
Rubén Eliécer Carvajalino C.

Coordinación editorial
Edwin Pardo Salazar

Corrección de estilo
Consuelo Cuesta Chiquillo

Diagramación y montaje de carátula
Jeferson Hernán Cuervo Sarmiento

Editorial UD
Universidad Distrital Francisco José de Caldas
Carrera 24 No. 34-37
Teléfono: 3239300 ext. 6202
Correo electrónico: publicaciones@udistrital.edu.co

Pérez Castillo, José Nelson
Introducción a la ciencia de datos en R : un enfoque práctico / José Nelson Pérez Castillo.-- Bogotá : Universidad Distrital Francisco José de Caldas, 2020.
178 páginas ; 24 cm. -- (Espacios)
Incluye referencias bibliográficas.
ISBN 978-958-787-171-5
1.Estadística – Fundamentos 2. R (Sistema para análisis estadísticos y gráficos) -- Estudio de casos I. Tít. II. Serie
519.5 cd 22 ed.
A1658438

CEP-Banco de la República-Biblioteca Luis Ángel

Contenido

Prefacio

Las grandes cantidades de datos originados por diversas fuentes, como los dispositivos móviles, las redes sociales, el internet de las cosas o las aplicaciones de nube, muchas de ellas con necesidades de análisis en tiempo real, han cambiado el contexto en el cual deben integrarse y analizarse los datos en las organizaciones. Este cambio ha dado origen a términos como *ciencia de datos, ingeniería de información, sistemas intensivos en datos, Big Data* o grandes volúmenes de datos.

Si bien los sistemas de información dentro de las organizaciones deben mantener su función de generar valor, diferenciación y mantener su oportunidad en la toma de decisiones, el análisis de datos moderno no está basado únicamente en técnicas y tecnologías clásicas para análisis estadístico, sino que requiere de nuevas tecnologías y metodologías para desarrollar soluciones propias para este tipo de información. Estos cambios afectan no solo las áreas técnicas, como el desarrollo de aplicaciones o las tecnologías de la información que deben utilizarse, sino también las franjas organizacionales a todos los niveles en formas que antes no eran factibles, a saber:

- A nivel estratégico y táctico de una organización, técnicas y tecnologías modernas para análisis de grandes volúmenes de datos permiten comprender y aprovechar la información propia y externa a la empresa, con el fin de entender los cambios y las tendencias del mercado, identificar opiniones de segmentos poblacionales relevantes para el negocio e interpretar los datos provenientes de redes sociales para generar análisis de competitividad.
- A nivel de desarrollo de aplicaciones se generan técnicas y metodologías propias para nuevos tipos de información, por ejemplo, información no estructurada que además es adaptable a diferentes campos de aplicación, permitiendo así el uso efectivo de los datos en el análisis de una problemática específica. Entre los campos de desarrollo se encuentran, entre otros: el análisis de comercio electrónico, el entendimiento en línea de la reacción de clientes frente a un producto o su competencia, la definición y el ajuste de políticas públicas, las

telecomunicaciones, los videojuegos en línea, las aplicaciones gubernamentales, aplicaciones de salud y ciencia, el análisis del comportamiento urbano y la predicción, prevención y reacción frente a desastres.

- A nivel de infraestructura de tecnologías de la información, tecnologías como *Hadoop* y bases de datos no relacionales (NoSQL) son utilizadas para facilitar la alta escalabilidad necesaria en el procesamiento y almacenamiento de este tipo de información. El uso de estas tecnologías, acompañado de la definición de arquitecturas orientadas a los datos, permite ofrecer sistemas robustos y eficientes que generan ventajas competitivas en los ámbitos empresarial, científico e investigativo.
- A nivel de información se debe trabajar tanto con fuentes estructuradas como no estructuradas de índole heterogéneas. La información proviene de fuentes autónomas, crece de forma exponencial y no es manipulable de forma efectiva con las herramientas tradicionales de gestión de bases de datos. Las fuentes suelen ser blogs, wikis, RSS, correos, comunidades participativas como las redes sociales y comunidades virtuales especializadas. Estas se integran con la información propia de las organizaciones y los individuos de manera ubicua. Se cuenta con ejemplos en dominios como biología, medicina, finanzas, análisis de mercados, imagen de personajes públicos, de comentarios en noticias de prensa y del estado de la comunidad a partir de los flujos de datos de redes sociales en tiempo real.

Objetivos y enfoque del libro

Este libro busca que el lector adquiera la capacidad de utilizar técnicas y herramientas de ciencia de datos para resolver problemas que involucren diversas fuentes de datos, datos estructurados y no estructurados, estando en capacidad de generar valor y diferenciación a las organizaciones, a través del análisis de sus conjuntos de datos. Busca, además, presentar, analizar y utilizar las oportunidades de innovación que ofrece la ciencia de datos en la toma de decisiones estratégicas y tácticas de una organización en el desarrollo de aplicaciones en diferentes campos del conocimiento y determinar cómo la plataforma R se presenta como alternativa especial para abordar estas oportunidades.

La naturaleza del libro es introductoria en tanto cubre los fundamentos de diversas temáticas dentro de la ciencia de datos con un enfoque más práctico que teórico. La bibliografía referenciada permitirá al lector ahondar en aspectos de su interés.

Este libro es uno de los resultados del proyecto de investigación titulado: *Modelos para el aporte eficaz de la ciencia de datos a la mejora de la competitividad del sector empresarial*

colombiano, cuyo principal objetivo es la aplicación de técnicas de análisis de datos modernas a la solución de problemáticas comunes en pequeñas y medianas empresas. Dentro de este estudio se determinó que un libro introductorio en idioma español, orientado tanto a técnicas de análisis de datos como en la plataforma R, será una herramienta valiosa para aquellas organizaciones que deseen emprender proyectos de ciencia de datos y que no cuenten con experiencia previa en estos aspectos o requieran de un punto de partida que las provea de un marco conceptual y de aplicaciones.

Público objetivo y prerrequisitos

El libro está pensado para servir de referencia en ciencia de datos aplicada, en particular aquellos que utilizan R como plataforma tecnológica. Partes del texto sirven de apoyo en cursos enfocados al análisis de grandes volúmenes de datos, sistemas de recomendación o minería de texto, entre otros, y puede utilizarse como autoaprendizaje o reforzamiento de los aspectos abordados. Para aprovechar de manera adecuada este texto, es necesario que el lector cuente con un conocimiento previo en las siguientes temáticas:

- Programación en un lenguaje de alto nivel, preferiblemente Java o Python.
- Fundamentos de bases de datos, en particular, conocimiento del lenguaje *Structured Query Language* (SQL). También puede ayudar tener experiencia con un sistema manejador de bases de datos relacional como PosgresQL, SQL Server, MySQL, Oracle u otros.
- Fundamentos de probabilidad y estadística a nivel de pregrado. Si bien el libro expone los conceptos estadísticos necesarios para análisis de datos, esta presentación debe considerarse más como un refuerzo de temas previamente dominados por el lector que como una exposición exhaustiva de la materia, la cual tomaría más espacio del disponible y se encuentra por fuera de los alcances del libro.

Organización del libro

El libro se ha dividido en un prefacio y seis capítulos con diversos tópicos. El capítulo 1 define la terminología básica a utilizar y presenta el ciclo de vida de un proyecto de análisis de datos. Los capítulos del 2 al 6 contienen material básico para análisis de datos en R de un contexto de datos a baja escala o de bajo volumen, y hace énfasis en el proceso denominado: *Análisis exploratorio de datos*; el capítulo 2, particularmente, introduce algunos modelos estadísticos básicos para análisis de datos; el capítulo 3, presenta los fundamentos de manipulación de objetos en R, así como la forma de implementar los modelos estadísticos incluidos en el capítulo anterior en la plataforma; el capítulo 4, alude a las fuentes de datos y el proceso de preparación de datos

en R; el capítulo 5, nos remite al aprendizaje de máquina en R, principalmente en los modelos de clasificación y segmentación; finalmente, el capítulo 6 muestra un caso de estudio en el cual se aplican técnicas de análisis de datos presentadas en las secciones anteriores.

Al finalizar el libro, el lector habrá asimilado una buena introducción a los distintos tópicos de la ciencia de datos y adquirido destrezas en el manejo de la plataforma R, incluyendo, por supuesto, el tener claro si esta plataforma es adecuada para el análisis de datos requeridos para su actividad profesional.

Introducción

Una locura es hacer la misma cosa una y otra vez esperando obtener resultados diferentes.
Si buscas resultados distintos, no hagas siempre lo mismo.

—Albert Einstein

Bienvenidos a "Introducción a la ciencia de datos en la plataforma R"

La temática de este libro es una herramienta relevante para un importante número de disciplinas, profesiones y organizaciones comerciales y no comerciales; basta con buscar el término *ciencia de datos* o *Data Science* en un motor de búsqueda como Google para obtener millones de resultados en menos de un segundo de procesamiento.

La naturaleza de este capítulo es introductoria, ofrece un panorama de la ciencia de datos y define los conceptos teniendo en cuenta que en otras fuentes bibliográficas es posible que algunos de estos términos tengan significados diferentes. El objetivo es que el lector conozca posibilidades y aplicaciones y esté en capacidad de desarrollar proyectos de ciencia de datos, en particular R, principal tema de este libro.

También se aborda el ciclo de vida de un proyecto de análisis de datos, tema fundamental para comprender, entre otras cosas, que un proyecto de este tipo debe contar con un objetivo de negocio claro, tener una naturaleza interdisciplinaria y gestionarse como un proyecto complejo, aplicando las mejores prácticas existentes en cada una de sus etapas. Una vez se logra claridad sobre este proceso, las tecnologías particulares, entre ellas la plataforma R, son herramientas de implementación a emplear en cada etapa; sin embargo, el dominio de la(s) tecnología(s) particular(es) a utilizar es solo una parte del espectro de habilidades involucradas en un proyecto de *ciencia de datos*.

1.1. Ciencia de datos y áreas relacionadas

Las grandes cantidades de datos originadas por diversas fuentes (dispositivos móviles, web, redes sociales, internet de las cosas, aplicaciones de nube), generadas a grandes velocidades, muchas de ellas con necesidades de análisis en tiempo real, han cambiado el contexto de integración y de análisis de datos en las organizaciones, dando origen a nuevos términos que de una u otra forma están relacionados con el

análisis y procesamiento de conjuntos de datos de diversa naturaleza. Algunos son términos nuevos para campos del conocimiento establecidos, mientras otros presentan cierta ambigüedad en su definición, a tal punto que algunos autores los consideran simplemente palabras "pegadizas", *buzzwords* o términos "de mercadeo".

La lista incluye términos como: ciencia de datos, minería de datos, analítica, inteligencia de negocios, aprendizaje de máquina, análisis exploratorio de datos, procesamiento analítico en línea u *On-Line Analytical Processing* (OLAP), analítica predictiva, analítica prescriptiva, minería de datos masivos y similares, incluyendo el cada vez más común *Big Data* o procesamiento de grandes volúmenes de datos. Es de notar que algunos de ellos ya existían mucho antes de instaurarse la ciencia de datos como un término. Esta sección es una reflexión sobre estos conceptos, para que el lector pueda asociar lo aprendido en este libro con el contenido de otras fuentes y referencias bibliográficas que utilicen otra terminología.

La idea central de la gran mayoría de términos es similar: los datos "puros" contienen *valor* y *conocimiento* que pueden y deberían ser descubiertos o extraídos, generalmente con el objetivo de contestar preguntas relevantes en una investigación o contexto determinado. Los datos cuentan una o varias historias y es labor del científico de datos, en un trabajo similar al realizado por un detective, lograr entender lo que tienen para contar.

Ahora bien, que los datos "puros" contengan valor y conocimiento ha llevado a que autores como [1] definan su manejo para los casos de *minería de datos, analítica predictiva, Big Data* y *ciencia de datos.*

1.1.1. Minería de datos, analítica predictiva, *Big Data* y ciencia de datos

En la minería de datos el problema a resolver es que, dado un conjunto de datos, incluyendo datos a gran escala, hay que descubrir patrones y modelos con las siguientes características [2]:

- Válidos: deben aplicar para nuevos datos a medida que estén disponibles.
- Útiles: debe ser posible actuar sobre ellos. Esto implica necesariamente que los modelos deben evitar el llamado principio de Bonferroni, esto es, ignorar que un conjunto de datos pueda tener características poco usuales que parecen importantes, pero que en realidad no lo son, llevando a análisis y conclusiones engañosas.
- Inesperados: deben ser no obvios al sistema o dentro del contexto del análisis.
- Entendibles: deben ser interpretables por los humanos.

Por *Analítica predictiva* se entiende la utilización de *métodos predictivos* en el análisis de datos, es decir, métodos que utilizan alguna(s) variable(s) para predecir el *comportamiento desconocido* o los *valores futuros* de otras variables [2]. Esta diferenciación es necesaria, pues existen otros métodos de análisis, como los *descriptivos* —patrones que interpretan el conjunto de datos— y los *confirmativos*, en los que se afirman o niegan supuestos sobre el conjunto de datos. Cabe mencionar adicionalmente que autores como [2] consideran a la analítica predictiva como un término equivalente a ciencia de datos, enfatizando en que las aplicaciones más interesantes de la ciencia de datos resaltan por la capacidad predictiva de sus modelos.

En el caso de los *Análisis exploratorios de datos* o *Exploratory Data Analysis* (EDA), se entienden todos aquellos métodos de descripción de un conjunto de datos en los que no se tiene ningún supuesto sobre la distribución estadística de estos antes de empezar el estudio, de ahí su carácter *exploratorio* [3]. Estos métodos de descripción tienen origen en que muchos de los proyectos de análisis de datos utilizaban "técnicas" *asumiendo previamente* que los datos se comportaban de una u otra manera, lo que conducía a resultados erróneos. Por el contrario, las técnicas exploratorias buscan, a través de la visualización y técnicas de estadística descriptiva, validar o descubrir estos supuestos en lugar de asumirlos como verdaderos.

Gran parte de la dificultad para definir con precisión cada término es la cantidad de perfiles o "culturas" que trabajan interdisciplinariamente en proyectos de análisis de datos. Por ejemplo, para un profesional especializado en bases de datos y sistemas de información, *minería de datos* es similar a lo que se conoce como *procesamiento analítico*, mientras que *Big Data* podría interpretarlo como *minería de datos masivos* [1].

De igual forma, un profesional en estadística (que suele contar con formación en aprendizaje de máquina) interpretaría *minería de datos* como *estadística inferencial aplicada*, *modelos inferenciales* o un sabor aplicado de *analítica predictiva*. La figura 1.1. ilustra un diagrama de Venn de algunos de los términos y afirma que *minería de datos* es la convergencia de sistemas de información, ciencias de la computación y estadística, lo cual no es para nada contradictorio con la definición del término previamente presentada.

Figura 1.1. Áreas del conocimiento involucradas en minería de datos

Fuente: Adaptada y traducida de [1]

En cuanto a la *inteligencia de negocios* o *Business Intelligence* (BO), se puede afirmar que es un término "sombrilla" bajo el cual se agrupan todas aquellas disciplinas que apoyan la *toma de decisiones* en el contexto particular de los negocios. Ciertamente la *minería de datos* aplicada a este contexto se considera como *inteligencia de negocios*, siempre que el proyecto de análisis de datos pueda determinar cómo se beneficia la organización de las predicciones hechas y cómo estas decisiones impactan todo lo demás, teniendo en cuenta que un objetivo común de un proyecto de análisis de datos en los negocios es *cambiar el comportamiento* del mismo en algún aspecto particular.

Junto con la *minería de datos* existen otras disciplinas que suelen encontrarse al hablar de *inteligencia de negocios*, y son la *Minería de procesos*, *Administración del conocimiento*, *Sistemas de reportes*, *Administración de procesos de negocio* o *Business Process Management* (BPM), *Gestión de proyectos de análisis* y otros tantos relacionados [2].

El último término sobre el cual se va a reflexionar es el que da título a este libro, el cual aún hoy no está exento de críticas y sigue presentando cierta polémica al respecto: se trata de la *ciencia de datos*. Su definición puede variar entre algunos autores como [5], que la concibe como la "[...] metodología de extraer conocimiento a partir

de los datos", un contenido que concuerda de alguna forma con la precisión conceptual anteriormente dada para la *minería de datos*.

En este libro se acogerá una de las definiciones más citadas: la presentada por [9], quien cita a Mason y Wiggins en su artículo "Una taxonomía de la ciencia de datos". Esta definición establece la ciencia de datos como una metodología o serie de pasos denominados *OSEMN*, a saber:

1. (O)btener los datos: proceso encargado de traer los datos desde otro lugar o generarlos, también implica identificar las fuentes de datos.
2. (S)*crub* o depurar los datos: proceso encargado de limpiar, filtrar, reemplazar y manejar valores no disponibles; convertir el formato de los datos antes de su análisis.
3. (E)xplorar los datos: proceso de *Análisis Exploratorio de Datos* (EDA).
4. (M)odelar los datos: proceso encargado de la creación de un modelo estadístico sobre los datos. Esto puede implicar la evaluación de varios modelos estadísticos y de aprendizaje de máquina, en cuyo caso se requiere dividir el conjunto de datos en datos de entrenamiento, entrenar el conjunto de datos y validar el modelo estimando el comportamiento del modelo con nuevos datos.
5. I(N)terpretar los datos: proceso en donde se definen las conclusiones de acuerdo con los datos, se evalúan los resultados y se comunican, generalmente utilizando visualizaciones.

Asimismo, se propone que los científicos de datos deben tener un conjunto de habilidades como afirma [1] citando a D. J. Patil, *Building Data Science Teams*, para desempeñar adecuadamente su labor:

- Experiencia técnica en alguna disciplina científica.
- Amplia curiosidad, entendida como el deseo de ir más allá de la superficie, descubrir y validar hipótesis que puedan ser probadas.
- Habilidad para contar una historia a partir de los datos y comunicarlos efectivamente.
- Habilidad para resolver problemas de forma diferente y creativa.

Al analizar en detalle la definición y las habilidades requeridas no debe resultar extraño por qué algunos autores consideran que la *ciencia de datos* no es más que un término de mercadeo para describir lo que hace siglos hacen los profesionales en estadística aplicada, por lo que *ciencia de datos* ≈ *estadística aplicada*. De hecho [5] menciona en su capítulo introductorio que la *ciencia de datos* es lo que anteriormente se llamaba "estadística" y que luego se conoció como *analítica de datos*.

Sin embargo, es importante resaltar que si bien la definición de ciencia de datos no es del todo clara respecto a cómo se diferencia este nuevo campo de la estadística, existen divergencias para considerar a la *ciencia de datos* como un nuevo campo distinto del anterior, a saber:

- Los profesionales en estadística suelen emplear datos de tipo numérico y no trabajan tradicionalmente la minería de texto o análisis de datos no numéricos. En ciencia de datos, el análisis de texto es fundamental y presenta oportunidades llamativas para aplicaciones.
- Usualmente los entregables en un proyecto de análisis de datos, desde el punto de vista de un profesional en estadística son *informes* o *reportes* con las conclusiones del estudio. Actualmente los entregables son sistemas de software funcionales e incluso los reportes siguen existiendo, pero usualmente como sistemas de software en sí mismos. Estos sistemas tienen como objetivo central el despliegue de modelos de decisión efectivos en *entornos de producción*. Cabe preguntarse si un profesional en estadística está entrenado para desarrollar software comercial y de producción, habilidad que sí se espera del profesional en ciencia de datos.
- Con el advenimiento de las tecnologías y metodologías para análisis de grandes volúmenes de datos, es necesario entender de algoritmos, arquitectura de computadores, plataformas de software y hardware, automatización de infraestructura tecnológica o *DevOps*, sistemas de almacenamiento y de información, procesamiento de imágenes, audio y vídeo, algoritmos en grafos, procesamiento en paralelo, sistema de automatización de flujos de datos, y en fin, un cuerpo de conocimiento que claramente no corresponde al núcleo básico del profesional en estadística.

Los dos últimos puntos son relevantes porque dejan entrever que los reportes tradicionales, tan válidos como pueden llegar a ser, ya no son el principal entregable de un proyecto de análisis de datos. En su lugar, se habla de un *producto orientado a datos* (*Product Data*), consumible por los usuarios como un nuevo sistema de software o aplicación resultado del proyecto.

Esta diferenciación no implica de ninguna manera que no se requieran las habilidades analíticas de los estadísticos: ¡nada más alejado de la realidad! También debe hacerse la salvedad que la anterior caracterización es de cierta forma una generalización, ya que es probable que existan profesionales en estadística con amplios conocimientos en los campos que usualmente no se les atribuye.

En la práctica, lo que sucede es que los proyectos de análisis de datos de cierta envergadura e impacto son de naturaleza *interdisciplinaria*, en donde las habilidades y los conocimientos en estadística son solo un subconjunto del total requerido. Otras

habilidades generalmente solicitadas son programación de computadores y conocimiento de dominio, ya que de otra forma no es posible validar si el conocimiento adquirido mediante análisis de datos es coherente con la problemática a resolver.

Para cerrar esta sección, es importante mencionar que con el tiempo habrá más especialización en lo referente a proyectos de análisis *que tienen que ver con grandes volúmenes de datos*, por lo que más términos irán surgiendo. A manera de ejemplo, ya se habla de *Ingeniería de datos* como el rol que desempeña un profesional en tecnologías de la información encargado de las plataformas hardware y software para análisis de grandes volúmenes de datos. Sin duda, habrá más que decir en un futuro sobre esta especialización, pero lo cierto es que, al día de hoy, la *ciencia de datos* ha emergido como una nueva disciplina en su propio derecho.

1.2. Ciclo de vida de un proyecto de análisis de datos

Un proyecto de análisis de datos está compuesto por varias etapas o procesos, los cuales cambian de acuerdo con los distintos enfoques. Sin embargo, antes de entrar a trabajar con los datos disponibles, es fundamental justificar el proyecto de análisis de datos, ya que debe recordarse que como resultado de un proyecto de análisis de datos se espera generar *transformación* en las organizaciones; por ejemplo, autores como [2] utilizan la expresión *conocimiento accionable*, luego, vale la pena preguntarse si el esfuerzo de crear un sistema, involucrar personal altamente calificado e interdisciplinario y otras labores tanto técnicas como administrativas que distan de ser triviales está plenamente justificado.

Como todo proyecto de naturaleza técnica, quizás la primera pregunta que debe hacerse es sobre *por qué* se está desarrollando el sistema. Lo más probable es que se tengan en mente algunos objetivos de negocio que se desean lograr, ya que salvo que se esté realizando un proyecto con fines netamente investigativos, no es racional emprender un reto de esta naturaleza sin algún objetivo de negocio claro o algún problema bien especificado que se desee resolver.

Una vez la visión de estos sistemas es clara, es posible contestar otras preguntas de naturaleza operativa, ¿cómo se va a utilizar el sistema?, o de naturaleza eminentemente técnica, ¿cuál es la información que se va a almacenar?, ¿cómo determinar que la implementación cumple con los objetivos de negocio propuesto? Sin embargo, es importante contar con la justificación del proyecto. Los siguientes casos enfatizan en esta idea:

- Ejemplo 1: una empresa de trenes emprendió un proyecto de análisis de datos para mejorar los procesos de mantenimiento de su flota, predecir las fallas y dar información de soporte actualizada y en tiempo real de los trenes. Si bien en este caso la justificación puede verse como una optimización de los procesos de la

empresa, es también importante señalar cómo el análisis predictivo de las fallas puede salvar vidas, lo cual tiene un gran impacto social y económico.

- Ejemplo 2: una empresa de consumo masivo determina que su publicidad no se está dirigiendo a las personas adecuadas, por lo cual emprende un proyecto de análisis de datos para mejorar el proceso publicitario perfilando de una mejor manera sus clientes. En este caso la justificación está orientada a incrementar las ventas mediante una publicidad mejor orientada.
- Ejemplo 3: una empresa de moda utiliza análisis de datos, procesamiento de imágenes y análisis de redes sociales para ofrecer servicios personalizados a sus clientes, ya que se determinó que para las personas es importante lucir "únicas", con prendas y accesorios totalmente personalizados y acorde con sus gustos y exigencias. Con esto la compañía busca no solo incrementar sus ventas, sino también generar diferenciación respecto a sus competidores.
- Ejemplo 4: un laboratorio científico adquiere millones de datos en tiempo real para comprobar una hipótesis científica con base en experimentos que generan grandes volúmenes de datos. Es de notar que en este caso el objetivo de negocio sigue estando claro, justificando el proyecto como parte de la misma misión del laboratorio.

Una vez se ha justificado el proyecto de acuerdo con la transformación del negocio deseada, es posible seguir las etapas del ciclo de vida del proceso hasta llevarlo a su fin. En la figura 1.2, las etapas del ciclo de vida se muestran en forma secuencial, pero en la práctica es común tener un proceso *iterativo*, en el que el resultado de una de las etapas puede llevar a reexaminar los resultados de una etapa anterior.

Figura 1.2. Ciclo de vida de un proyecto de análisis de datos

Fuente: Tomada y traducida de [1]

Si bien las etapas del ciclo de vida de un proyecto de análisis de datos varían entre los autores, es común encontrar las siguientes [1]:

1. Planeación y definición del problema.
2. Preparación de datos.

3. Análisis de datos.
4. Despliegue.

La primera etapa, *Planeación y definición del problema*, es la encargada de la gestión operativa y los análisis propios de un proyecto; en esta, se definen los entregables del proyecto, se ensamblan los equipos de desarrollo y análisis, se realizan los análisis de costo/beneficio y una vez se tiene una visión clara del problema a resolver, se generan los factores que determinarán el éxito del proyecto.

En la segunda etapa, *Preparación de datos*, se realizan los procesos que modifican los datos antes de su análisis. Esto incluye, acceder a los datos, posiblemente combinarlos con otros, realizar un análisis de la calidad de los datos, realizar transformaciones en los datos y segmentarlos.

En la tercera etapa, de *Análisis* propiamente dicho, se busca explorar relaciones entre los datos tratando de identificar aquellas relaciones y tendencias no triviales. En esta etapa se elaboran los modelos estadísticos (resumen de datos o modelos de regresión) o modelos de aprendizaje de máquina (modelos de clasificación o de análisis de grupo).

Finalmente, en la etapa de *Despliegue*, se aplica la solución en el negocio con el objetivo de generar *cambi*o, esta puede ser un reporte, un sistema de software (aplicación), alguna herramienta de integración con otras herramientas de apoyo a la toma de decisiones o alguna directiva que implique cambio en los procesos de la organización.

Para concluir esta sección, es importante resaltar que existen otras definiciones de ciclos de vida de un proyecto de datos. En particular, es de interés la metodología ágil aplicada al análisis de datos [17]. Se invita al lector interesado en este tema a revisar las fuentes bibliográficas suministradas al final del capítulo.

1.3. Sistemas intensivos en datos y *Big Data*

Dentro del cuerpo de conocimiento y las habilidades técnicas asociadas a un científico de datos, cada vez cobra más relevancia la capacidad para tratar con *sistemas intensivos* en datos, bien sea para construirlos o para analizar la información contenida en ellos, información que suele categorizarse como *Big Data*.

Si bien este es un libro introductorio de análisis de datos utilizando técnicas de pequeña escala, para un científico de datos es fundamental entender a qué se hace referencia cuando se habla de *Big Data*, cómo luce un sistema intensivo en datos, y quizás más relevante, cómo afecta el hecho de ser *Big Data* las técnicas de análisis de datos de este libro. Es en este sentido que esta sección ha sido agregada al texto,

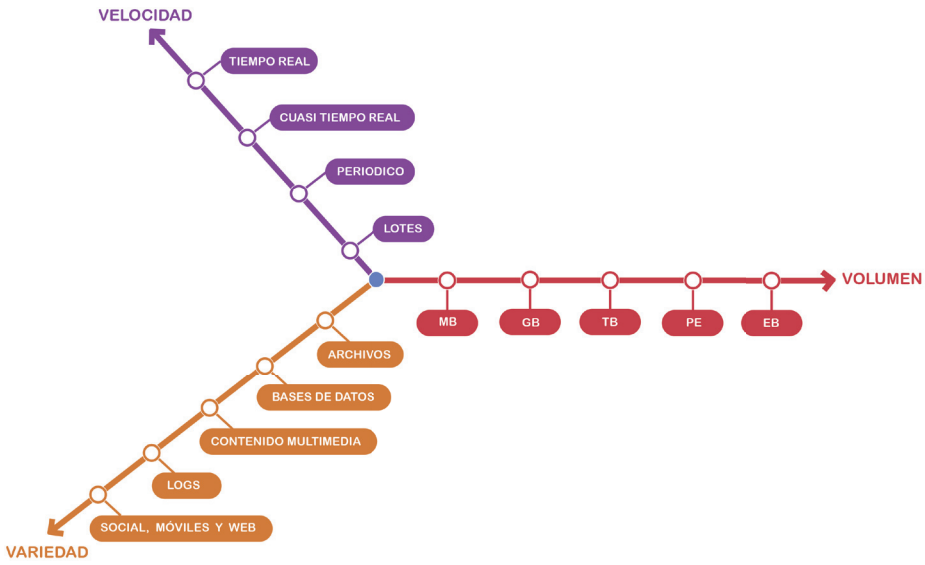
quizás más como un "abrebocas" a esta importante temática, ya que los detalles técnicos de análisis de *Big Data* o diseño de sistemas intensivos en datos pueden requerir un libro entero.

Se denominan grandes volúmenes de datos, datos a gran escala o *Big Data* a conjuntos de datos cuyo tamaño y complejidad derivan en que este conjunto no pueda ser procesado por herramientas tradicionales como los gestores de bases de datos relacionales. Debido a su tamaño y complejidad, las herramientas tradicionales tienen problemas para adquirir, capturar, mover, buscar, almacenar y procesar estos conjuntos de datos en un tiempo o costo razonable. Si bien en un comienzo se hacía referencia explícita al tamaño del conjunto de datos (1 PB) para determinar si clasificaba como un volumen de datos a gran escala, actualmente esto no es del todo preciso (además de no ser sostenible en el tiempo), por esto se evitará hacer uso de tal "definición" en este libro. Con esto se quiere decir que no es del todo preciso decir que *Big Data* es a partir de "x" terabytes, petabytes o exabytes, si bien es común encontrar conjuntos de datos de ese tamaño. Como se determinará posteriormente, muchas veces otras características del conjunto de datos generan mayores dificultades de análisis que el tamaño mismo.

La definición comúnmente encontrada en la literatura es debida a Douglas Laney, quien a la fecha de la escritura de este libro trabaja para la consultora Gartner y estableció "las 3 Vs" que definen los grandes volúmenes de datos, a saber:

- Volumen: esta característica hace referencia al tamaño del conjunto de datos, expresado en bytes. Algunas unidades del conjunto de datos pueden ser mega-bytes (MB), gigabytes (GB), terabytes (TB), petabytes (PB), o exabytes (EB); en un futuro no muy lejano se hablará incluso de zettabytes (ZB) o yottabytes (YB).
- Velocidad: esta característica hace referencia a la tasa de llegada de los datos y qué tan rápido necesitan ser procesados. Puede variar desde un procesamiento en lote que se ejecute cada cierto periodo hasta la necesidad de procesar el conjunto de datos en tiempo real.
- Variedad: esta característica hace referencia a la forma en que se presenta el conjunto de datos, siendo posible que este corresponda a datos estructurados, texto no estructurado, datos de sensores, información de fuentes multimedia o logs de dispositivos, entre otros. Generalmente, los datos no estructurados no vienen listos para la integración con las aplicaciones, luego, deben procesarse de forma diferente.

Figura 1.3. Las "3 Vs" que definen *Big Data*: volumen, velocidad y variedad



Fuente: Adaptado y traducido de [6]

Algunos autores, como por ejemplo [6], proponen que la definición de *Big Data* debería expandirse con otras "3 Vs", a saber:

- Visión: esta característica hace referencia a que todo proyecto de análisis de grandes volúmenes de datos debe tener un propósito y un plan.
- Veracidad: hace referencia a que los datos (en particular, aquellos que se pueden considerar como una fuente de grandes volúmenes de datos) deben estar conformes con un conjunto de especificaciones.
- Validación: hace referencia a que las conclusiones a las que llegue un proyecto de análisis de grandes volúmenes de datos debe satisfacer el propósito del proyecto.

Cabe mencionar que la característica de *veracidad* es comúnmente encontrada en las definiciones de *Big Data*, convirtiéndose en la cuarta "V".

Los grandes volúmenes de datos plantean nuevos retos y posibilidades tanto para las organizaciones como para los profesionales de tecnologías de la información y analistas de datos cuyos procesos, tecnologías, herramientas de análisis, posibilidades de negocio, y en general, aspectos a varios niveles, se ven directamente afectados por las características del conjunto de datos. Por ejemplo, si solo se tuviese en cuenta la "V" relativa al *volumen*, una organización o analista determinaría rápidamente que

el tamaño mismo del conjunto de datos impacta la captura, transferencia, almacenamiento, procesamiento, presentación, análisis y latencia (tiempo que toma acceder a un dato), por lo que probablemente sus sistemas de información actuales no estén en capacidad de manejar adecuadamente un conjunto de esta naturaleza. Las consecuencias inmediatas de lo anterior plantean, sin lugar a dudas, nuevos retos desde el punto de vista tecnológico, de análisis, de metodologías y procesos, pero también estos retos traen consigo enormes posibilidades para el desarrollo de las organizaciones, los individuos y la sociedad en general.

Desde el punto de vista técnico, quizás el cambio más importante para los sistemas de información es que al analizar grandes volúmenes de datos *hay que llevar el procesamiento a los datos*, no los datos al procesamiento. Esto es debido a que el volumen del conjunto de datos no es posible almacenarlo en memoria principal antes de procesarlo, y este es el enfoque que las aplicaciones tradicionales utilizan; lo anterior sin tener en cuenta el costo de mover los datos hacia el procesamiento —hablando en términos de tiempo—. Como es de esperarse, llevar el procesamiento a los datos y no al contrario implica un conjunto de herramientas totalmente nuevo, así como modelos y técnicas tanto de análisis como de almacenamiento diferentes a las utilizadas en un contexto de baja escala.

En particular, es mandatorio a la hora de procesar y almacenar estos conjuntos pensar en términos de *procesamiento paralelo*, ya que es la única forma de obtener un buen desempeño de los sistemas que tratan con estos conjuntos de datos y de *escalabilidad* para poder soportar la cada vez más creciente cantidad de datos sin tener que rediseñar los sistemas. Este hecho impacta las organizaciones a todos los niveles; por ejemplo, tradicionalmente las organizaciones tenían sus sistemas de información transaccionales en un único servidor de amplias prestaciones; el ser un único servidor facilitaba enormemente las labores de mantenimiento y tolerancia a fallas, y mantenía bajo control los costos de licenciamiento.

Al hablar de un sistema de almacenamiento y procesamiento de grandes volúmenes de datos se habla de procesamiento en paralelo y escalabilidad, con lo cual el aspecto de licenciamiento, el manejo de fallas y el simple hecho de determinar cuántos y cómo deben ser los nuevos servidores afecta las operaciones diarias de la organización en lo referente a sus tecnologías y sistemas de información.

Como se mencionó anteriormente, todo reto trae consigo muchas posibilidades. Inicialmente se podría hacer mención a la de almacenar tantos datos como se pueda (incluidos datos históricos), ya que con las tecnologías actuales de almacenamiento de datos y sus restricciones es necesario ser en exceso escrupulosos sobre los datos que deben almacenarse con el objetivo de reducir su volumen y facilitar su análisis.

La serie de libros *Big Data Now* [12, 16] hace referencia a la frase "Cuando puedas, quédate (almacena) con todo" como un principio fundamental de *Big Data* que está en línea con la anterior afirmación. Esto cambia la forma de pensar de las organizaciones que pueden proceder a capturar, primero, los datos, y realizar las preguntas después, lo cual es sustancialmente diferente al filtro de datos que se hacía anteriormente. El almacenar tantos datos como sea posible, incluso de diversas fuentes, y poder procesarlos en un tiempo factible, ofrece una potencial ventaja competitiva sobre aquellos que no cuentan con todo el espectro de información a nivel temporal o de fuentes.

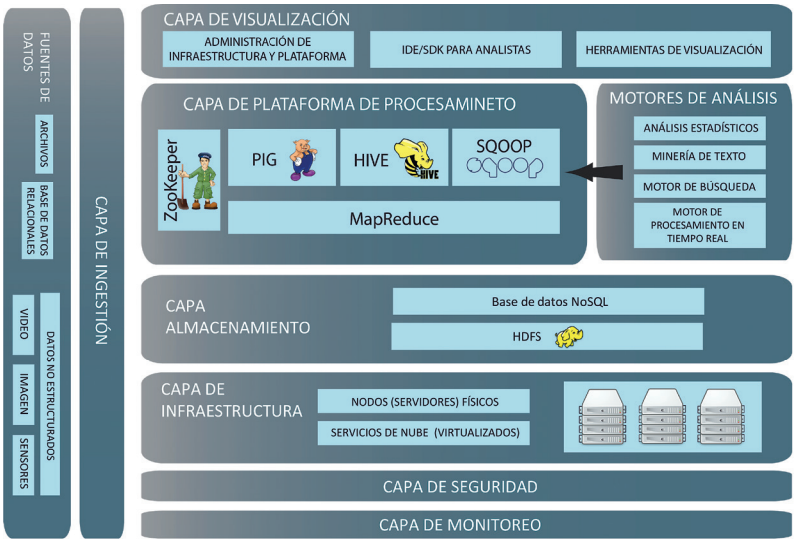
También es cierto que *Big Data* es un habilitador de nuevos negocios, servicios y productos, como lo reflejan casi a diario servicios como Google, Facebook o Amazon. En esta misma referencia —entre otras muchas fuentes— se encuentran gran variedad de casos de negocio y cómo las organizaciones han aprovechado las tecnologías para análisis, almacenamiento y procesamiento de grandes volúmenes de datos para solucionar problemas complejos (que no eran posibles anteriormente con sus herramientas tradicionales) o para satisfacer de mejor forma sus objetivos misionales, alinear sus estrategias e iniciativas, crear nuevos productos o servicios o realizar mejores pronósticos de aspectos que afectan el negocio. Mención aparte merecen todas las aplicaciones de las tecnologías que están por ser desarrolladas, ya que el verdadero potencial del *Big Data* parece que hasta ahora está por verse.

1.4. Componentes de un sistema *Big Data*

Las soluciones de almacenamiento, procesamiento y análisis de grandes volúmenes de datos, simplificadas en este libro como *Sistemas Intensivos en Datos*, soluciones *Big Data* o sistemas *Big Data*, son soluciones multicomponentes, compuestas de infraestructuras heterogéneas, bases de datos con varias tecnologías, herramientas de visualización y análisis de varios fabricantes, entre otros componentes.

Debido a la heterogeneidad propia de un sistema de este tipo, puede que no sea posible establecer una única arquitectura o estructura, por lo que es necesario restringirse a aquellos componentes más comúnmente encontrados, que en la mayoría de veces son requerimientos indispensables para que el sistema se considere de grandes volúmenes de datos. Una arquitectura común y ampliamente referenciada en la literatura para un sistema *Big Data* se muestra en la figura 1.4.

Figura 1.4. Arquitectura de un sistema *Big Data*



Fuente: Adaptada y traducida de [8]

Según se observa en la figura, entre estos componentes frecuentemente hallados en un sistema *Big Data* se encuentran:

1. Un método para mover datos hacia y desde el sistema en forma segura, sin pérdida y lo más eficiente posible. Esto puede implicar mover los datos hacia y desde archivos ya existentes o un sistema gestor de bases de datos o de apoyo al procesamiento/almacenamiento u otras fuentes. En general, este proceso es conocido como *ingestión de los datos* y es común encontrar herramientas dedicadas para tal fin.

2. Un sistema de almacenamiento distribuido en varios servidores, escalable a miles de servidores, con redundancia en los datos en caso de fallos de hardware. Además, debe ser económicamente viable. Este sistema de almacenamiento debe permitir el procesamiento local en donde residen los datos, característica diferenciadora del análisis de grandes volúmenes de datos. Usualmente el sistema de almacenamiento se divide en la *capa de infraestructura* de tecnología física (granjas de servidores físicos o virtualizados en una solución de nube) y la *capa de almacenamiento* en donde reside el sistema de archivos distribuido y los gestores de bases de datos NoSQL (también con características de distribución, escalabilidad y con tolerancia a fallas, incluidas en su mismo diseño). Un ejemplo es el sistema de archivos *Hadoop File System* (HDFS), inicialmente un proyecto de la empresa Yahoo ahora convertido en un proyecto de la Apache Software Foundation.

3. Una *plataforma de procesamiento*, esto es, un conjunto de herramientas amplio para procesamiento en paralelo con los datos directamente ingresados sobre la capa de almacenamiento.
4. Motores especializados en análisis estadístico, minería de texto, motores de búsqueda e indexadores o motores de procesamiento en tiempo real. Estos motores especializados usualmente funcionan de manera cooperativa con la plataforma de procesamiento.
5. Sistemas de visualización y reporte.
6. Un método para que se pueda acceder de forma programática al sistema mediante un entorno de desarrollo (IDE) o kit de desarrollo (SDK).

En entornos reales de operación, más allá de una prueba de concepto, es posible contar con sistemas de monitorización de la infraestructura, una capa de seguridad, herramientas de automatización de tareas y administración de toda la plataforma, así como herramientas adicionales para procesamiento de datos que permitan interoperar con otros sistemas de información ya existentes.

Como parte de la denominada *inteligencia de negocios* suele encontrarse el término *Bodega de datos* que es un repositorio central *desnormalizado*, el cual es alimentado por los sistemas transaccionales con el objetivo de realizar análisis —usualmente *fuera de línea*— para el apoyo de la toma de decisiones. De esta breve descripción se pueden establecer someramente las diferencias entre una bodega de datos y una solución *Big Data* [8]:

- En *Big Data* los datos se almacenan en servidores distribuidos en lugar de un servidor central.
- En *Big Data* las funciones de procesamiento se llevan a los datos en lugar de llevar los datos al procesamiento.
- En *Big Data* los datos son de varios formatos, tanto estructurados como no estructurados.
- En *Big Data* los datos se pueden procesar tanto en tiempo real como fuera de línea.
- En *Big Data* las tecnologías están basadas, en su mayoría, en conceptos de procesamiento paralelo.

Esto no excluye que una *bodega de datos* no pueda ser parte de un sistema *Big Data*, y de hecho se encuentran en la literatura arquitecturas en las que una bodega de datos se integra a un sistema *Big Data,* como la que ilustra la figura 1.5. En esta arquitectura, la cual es una variación de la mostrada en la figura 1.4, se parte de la información

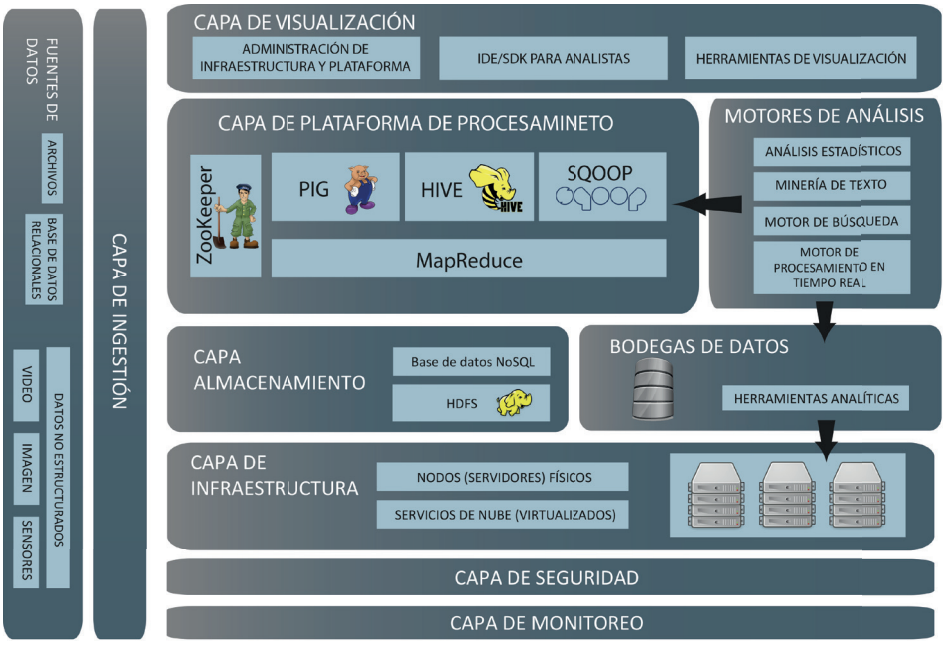
analizada o generada por los motores de análisis, que es almacenada en la bodega, lo cual no excluye que la misma bodega de datos sea utilizada como una fuente de información que alimenta la capa de almacenamiento de la solución *Big Data*. Parece ser que las tecnologías actuales involucran tantos conceptos comúnmente asociados a bodegas de datos en soluciones *Big Data* que puede que en un futuro se presente cierta convergencia o equivalencia de los términos.

Al notar que cada uno de los componentes de un sistema *Big Data* puede ser en sí mismo un sistema complejo, cabe la duda respecto a la necesidad de un sistema de este tipo, sobre todo cuando las herramientas de gestión de sistemas de bases relacionales cuentan con un alto nivel de madurez, robustez y eficiencia. Esta duda surge sobre todo si se tiene en cuenta que es común encontrar en la literatura la frase "pueden abordarse problemas que no eran posibles con *Relational DataBase Management Systems* (RDBMS) o Sistemas Gestores de Bases de Datos Relacionales" a la hora de hablar de los beneficios de un sistema para procesamiento de grandes volúmenes de datos.

El problema principal de un sistema gestor de bases de datos (RDBMS) es que está basado en conceptos postulados hace más de 40 años para datos almacenados en un formato estructurado con infraestructura centralizada en su mayoría. Los datos eran generados por organizaciones y empresas para conducir análisis dentro de las mismas. Con el advenimiento de internet global, las redes sociales y los dispositivos móviles, ya no son solo las organizaciones las que generan datos, sino que los individuos y los mismos dispositivos también contribuyen con datos, que distan de estar estructurados y son variados en su naturaleza.

Para estos datos, los sistemas gestores de bases de datos son ineficientes, rígidos y costosos, ya que no están diseñados con este fin, razón por la cual grandes corporaciones diseñaron nuevas tecnologías y conceptos que tratan estos datos voluminosos, variados y generados a gran velocidad de una forma más eficiente y a un menor costo. A propósito del costo, es importante resaltar que el análisis de grandes volúmenes de datos estaba al alcance de grandes corporaciones, pero a un costo elevado. Hoy por hoy, debido en gran parte al surgimiento de la computación en la nube, estos análisis son factibles incluso para pequeñas *startups*.

Figura 1.5. Arquitectura de un sistema *Big Data* incorporando bodegas de datos



Fuente: Adaptada y traducida de [8]

Referencias bibliográficas

[1] J. Leskovec, A. Rajaraman y J. Ullma, *Mining of Massive Datasets*. Second Edition: Cambridge University Press, 2014. [En línea]. Disponible en: https://stanford.io/2NXweOP ^3,5,7,12

[2] J. Ledolter, *Data Mining and Business Analytics with R*. New Jersey: John Wiley & Sons, 2013. [En línea]. Disponible en: https://stanford.io/2NXweOP ^4,6,10

[3] W. L. Martinez, A. R. Martinez y J. L. Solka, *Exploratory Data Analysis with MATLAB*. Third Edition, Boca Ratón Florida: Editorial CRC Press Taylor & Francis Group, 2017. [En línea]. Disponible en: http://bit.ly/32udycH ^4

[4] E. Mayor, *Learning Predictive Analytics with R*. Packt Publishing, 2015. [En línea]. Disponible en: http://bit.ly/2G8RFpJ

[5] M. Pathak, *Beginning Data Science with R*. Springer International Publishing, 2014. [En línea]. Disponible en: http://bit.ly/2XT0bPA ^6,8

[6] J. J. Berman, *Principles of Big Data: Preparing, sharing and analyzing complex information*. Second edition: Academic Press, 2013. [En línea]. Disponible en: https://amzn.to/2SgS8ee ^15

[7] G. J. Myatt y W. P. Johnson, *Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining*. Second Edition: John Wiley and Sons Inc, 2014. [En línea]. Disponible en: https://amzn.to/2GeYIwY

[8] N. Sawant y H. Shah, *Big Data Application Architecture Q&A. A Problem-Solution Approach*. Apress, 2013. [En línea]. Disponible en: https://amzn.to/2NWolsY

[9] J. Janssens, *Data Science at the Command Line*. O'Reilly Media Inc., 2015. [En línea]. Disponible en: http://bit.ly/2xNiNGb ^6

[10] D. Feinleib, *Big Data Bootcamp*. Apress, 2013.

[11] P. Warden, *Big Data Glossary*. O'Reilly Media Inc., 2011. [En línea]. Disponible en: http://bit.ly/2YWLCfo

[12] O'Reilly Media, Inc. *Big Data Now*. O'Reilly Media Inc., Edition 2016. [En línea]. Disponible en: https://oreil.ly/2XOMCkh ^17

[13] O'Reilly Media, Inc. *Big Data Now*: 2015 Edition, O'Reilly Media Inc., 2015.

[14] O'Reilly Media, Inc. *Big Data Now*: 2014 Edition, O'Reilly Media Inc., 2014.

[15] O'Reilly Media, Inc. *Big Data Now*: 2013 Edition, O'Reilly Media Inc., 2013.

[16] O'Reilly Media, Inc. *Big Data Now*: O'Reilly Media Inc., Edition, 2012. [En línea]. Disponible en: https://oreil.ly/2GhdyDo ^17

[17] R. Jurney, *Agile Data Science.* O'Reilly Media Inc., 2014. [En línea]. Disponible en: http://bit.ly/2LViQrJ ^13

2. Principios de estadística

Todos los modelos están equivocados, pero algunos modelos son más útiles que otros.
—George Box

El propósito de este capítulo es dar una introducción a los conceptos estadísticos necesarios para realizar un análisis exploratorio de un conjunto de datos. El enfoque, como se mencionó en la introducción del libro, es de aplicación y no hará énfasis en teorías matemáticas ni estadísticas subyacentes, las cuales pueden ser revisadas por el lector en los textos indicados en las referencias. El capítulo aborda tres temáticas: la sección 2.2, presenta una introducción referente a estadística descriptiva, en particular lo concerniente a las estadísticas de resumen del conjunto de datos; la sección 2.3, es una introducción a los temas que usualmente se tratan bajo la etiqueta estadística inferencial, esto es, pruebas de hipótesis y los conceptos básicos de modelos lineales. Finalmente, la sección 2.4, es una introducción al análisis estadístico de series de tiempo, reconociendo que muchos de los análisis encontrados en la práctica involucran el componente temporal, de ahí la relevancia de estudiar estos modelos.

2.1. Observaciones y variables

El punto de partida de todo análisis estadístico es, naturalmente, un conjunto de información que contiene datos medidos o recolectados de algún aspecto, hecho o experimento particular; estos datos pueden encontrarse en formato numérico o alfanumérico, dependiendo de la naturaleza de lo que se está caracterizando. El objetivo de un buen análisis estadístico es, teniendo en cuenta las limitaciones propias de los datos, obtener la mayor información posible de ese conjunto de datos, bien sea para apoyar la toma de alguna decisión, realizar un pronóstico de algo en particular o realizar una descripción precisa de algún fenómeno, entre otros fines que justifican el análisis del conjunto de datos.

La representación más común de este conjunto de datos es un formato conocido como data frame o *data table* (tabla de datos). Esta representación tiene un formato o estructura en la que las *filas* representan ítems individuales conocidos como *observaciones* y cada *columna* representa distintos atributos o *variables* de cada observación. Se

define una *variable* como el conjunto de valores de un atributo que describe aspectos a través de todas las observaciones.

En forma genérica, un data frame puede verse de la siguiente forma (tabla 2.1), en donde cada o_i representa una observación y cada x_i una variable.

Tabla 2.1. Estructura de un data frame

	x_1	x_2	x_3	$\cdots$	x_p
o_1	x_{11}	x_{12}	x_{13}	$\cdots$	x_{1p}
o_2	x_{21}	x_{22}	x_{23}	$\cdots$	x_{2p}
o_3	x_{31}	x_{32}	x_{33}	$\cdots$	x_{3p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_n	x_{n1}	x_{n2}	x_{n3}	$\cdots$	x_{np}

Fuente: Elaboración propia

Que la información deba estar en o pueda llevarse fácilmente a este formato no debe menospreciarse, pues esta restricción de formato o *estructura* puede que no sea aplicable a muchas fuentes de datos *no estructuradas*, comúnmente utilizadas en el análisis de grandes volúmenes de datos. Ejemplos particulares de conjuntos de datos se muestran a continuación.

- Ejemplo 1. En el contexto de un estudio clínico en una organización enfocada en investigación en salud es posible obtener un conjunto de datos con los datos recolectados de un paciente: nombre, peso, tipo de sangre, altura, edad, enfermedades previas, enfermedades de familiares y otros. En este conjunto de datos cada paciente representa una *observación* y cada dato que de él se recolecta representa una *variable*.
- Ejemplo 2. En la industria automotriz es posible tener un conjunto con datos sobre vehículos, que son las *observaciones* del conjunto. Cada vehículo puede describirse en términos de su fabricante, marca, modelo, cilindraje, número de cilindros, eficiencia de gasolina, tipo de motor, autonomía, tiempo de aceleración, peso y otras tantas *variables*.
- Ejemplo 3. En las tiendas comerciales es posible que el conjunto de datos contenga información de las diferentes transacciones llevadas a cabo por los consumidores. Cada transacción (*observación*) puede estar caracterizada por *variable*s como la fecha de la transacción, producto adquirido, cantidad, marca, costo unitario, costo total y otros atributos que caracterizan la transacción.

Contar con un conjunto de datos con muchas observaciones o múltiples variables dificulta el proceso de identificar tendencias o relaciones entre las variables; debido a ello, es recomendable una primera caracterización *individual* de cada variable, lo cual puede hacerse apelando a medidas de tendencia central y a la visualización.

Sin embargo, las medidas de tendencia central no aplican para todo tipo de variable; por ello, es fundamental reflexionar sobre los distintos tipos de variables que pueden encontrarse en una tabla de datos o data frame, aclarando que la clasificación varía de acuerdo con su *naturaleza* o *continuidad*, la cual debe estar en función de su *escala* y su *rol en el modelo*.

De acuerdo con su *naturaleza o continuidad* es común clasificar las variables en *continuas o categóricas*; las continuas, como su nombre lo indica, son variables numéricas que pueden tomar un número infinito de valores reales. En contraparte, las variables categóricas o discretas toman únicamente valores dentro de un conjunto finito; estos valores se denominan *niveles*. Ejemplos de variables continuas son peso, aceleración, ventas, altura, densidad poblacional y área. Por otro lado, en variables categóricas pueden encontrarse: género (conjunto de dos o más valores), colores, días de la semana, meses del año, idiomas, tipo de estudio, nivel académico y otras, según se establece en la definición.

Los *conteos*, variables que toman valores enteros y representan cuántas veces algún hecho ocurrió, también se consideran variables categóricas debido a su naturaleza discreta. Desde el punto de vista estadístico es necesario tener en cuenta este tipo de variable, ya que algunos análisis estadísticos —en particular, las regresiones lineales— no son apropiados para las variables de conteo.

De acuerdo con su *escala*, las variables pueden clasificarse en:

- Nominales: tienen un número limitado de valores no ordenables, puesto que el ordenamiento no tiene ningún significado. La mayoría de variables categóricas pertenecen a este tipo.
- Ordinales: aquella con un número limitado de valores que pueden ser *ordenados* de acuerdo con algún criterio que permite establecer un *escalafón* entre los valores. Ejemplo común de este tipo de variable es aquella que puede tomar los valores "Muy Alto", "Alto", "Bajo" y "Medio"; como se observa, es un conjunto limitado de valores, pero es posible ordenarlos de forma que "Muy Alto" tenga una mejor posición en un escalafón que "Bajo". Debe notarse que la *diferencia* entre estos valores no puede ser determinada, ya que no tendría sentido comparar la diferencia entre "Alto" y "Medio", con la diferencia entre "Medio" y "Bajo". Otro ejemplo de este tipo de variables es la clasificación de las quemaduras en "1er Grado", "2do Grado" y "3er Grado". En este contexto, se determina que "3er Grado" es de mayor gravedad que las anteriores, pero una

vez más la diferencia no tiene sentido, pues no se puede determinar si la diferencia entre "3er Grado" y "2do Grado" es mayor, menor o igual que la diferencia entre "2do Grado" y "1er Grado".

- Intervalos: variables en las cuales las diferencias de valores pueden ser comparados. Este tipo de variable no es común, siendo el ejemplo estándar las escalas de temperatura; en este caso, una diferencia entre 28° y 23° (diferencia de 5°) representa el mismo cambio de temperatura que la diferencia entre 17° y 12°. Sin embargo, la multiplicación y la división en este tipo de variables no tiene mucho sentido, puesto que no es cierto que 20° sea "el doble de caliente" que 10°.
- Ratios o proporciones: variables en las que tanto las diferencia de valores (intervalos) como las proporciones pueden ser comparados. Ejemplos de este tipo de variable son medidas físicas como altura, peso u otras como dinero, ya que tiene sentido decir que alguien con $1.000 tiene 10 veces más dinero que alguien con $100, o que alguien que pesa 270 kg pesa tres veces más que alguien que pesa 90 kg.

De acuerdo con su *rol en el modelo*, las variables pueden clasificarse en *independientes*, también denominadas explicativas, predictoras o de *respuesta*. En la variable respuesta se busca entender el comportamiento estadístico. Bajo este enfoque, el objetivo del análisis es *explicar* cómo variaciones en las variables independientes se asocian con variaciones de la variable respuesta.

Una vez se determina el tipo de cada variable es posible iniciar el proceso exploratorio de datos obteniendo indicadores que o bien describan las variables del conjunto de datos o bien permitan realizar predicciones o generalizaciones de los valores de estas variables. Estos indicadores se denominan *estadísticas* y son el tema de las siguientes secciones.

2.2. Fundamentos de estadística descriptiva 2.0

Estadística descriptiva hace referencia a estadísticas que describen de forma resumida características del conjunto de datos o colección de información. Su objetivo es resumir muestras de datos de manera cuantitativa (como las llamadas estadísticas de resumen) o de forma visual mediante gráficas sencillas de entender. Estas sirven como punto de partida para un análisis estadístico más exhaustivo.

2.2.1. Medidas de tendencia central

Las primeras estadísticas usualmente encontradas en estadística descriptiva son las *medidas de tendencia central*, las cuales buscan caracterizar de forma cuantitativa el valor alrededor del cual fluctúa una variable de forma que este valor se pueda

considerar como el centro de los valores de la variable. Las tres principales estadísticas para calcular la tendencia central son la *moda*, la *media* y la *mediana*.

La *moda* es el valor que más frecuentemente ocurre en un variable. Está definida para variables tanto numéricas (continuas o conteos) como para variables categóricas. La *media* puede definirse de diferentes maneras, siendo la más comúnmente usada la *media aritmética* ($\bar{y}$). Esta se define como la suma de todos los valores de una variable, dividida entre el número de valores disponibles. Matemáticamente, esto se expresa como:

$$\overline{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{2.1}$$

A diferencia de la *moda*, la *media* está definida únicamente para variables de naturaleza numérica. Una característica que merece la pena mencionar es que, lamentablemente, la media aritmética es sensible a valores extremos u *outliers*, por lo cual un único valor en extremo grande o pequeño respecto al resto de valores tendrá un efecto considerable en el cálculo.

Para procesos que no presentan una característica aditiva, sino multiplicativa, existe una medida de tendencia central que se considera más apropiada que la media aritmética, al no ser tan sensible a los valores extremos. Esta medida, denominada *media geométrica*, se define como sigue:

$$\hat{y} = \sqrt[n]{\prod_{i=1}^{n} y_i} \tag{2.2}$$

La *mediana* es una medida de tendencia central que se define como el valor que se encuentra exactamente en la mitad de la colección *después de ordenar ascendentemente la variable*. Esta definición no presenta ningún problema si el número de datos disponibles es impar, pero debe ser adaptada si el número de datos es par. En este caso, se define la *mediana* como el promedio de los dos valores que se encuentran en la mitad. Cabe mencionar que a diferencia de la media, la mediana no es sensible a los datos extremos.

- Ejemplo 4: se desea calcular la *moda*, la *media* y la *mediana* de los siguientes valores: 10, 12, 9, 9, 14, 13, 8, 10, 12, 12, 13, 10, 9, 14, 10.

La *moda* del anterior conjunto de valores es 10, pues es el valor que más veces aparece: 4.

La *media* aritmética se calcula utilizando la definición:

$$\bar{y} = \frac{10+12+9+9+14+13+8+10+12+12+13+10+9+14+10}{15} = 11 \tag{2.3}$$

Para calcular la mediana es necesario ordenar ascendentemente la variable. El resultado de este proceso es: 8, 9, 9, 9, 10, 10, 10, 10, 12, 12, 12, 13, 13, 14, 14. El valor que se encuentra en la mitad de esta secuencia (en la posición 8) es 10, valor correspondiente a la mediana de los valores.

En este ejemplo cabe mencionar que: el que la moda y la mediana tengan el mismo valor debe considerarse más como un hecho fortuito que como una regla general. Sin embargo, al ser las tres estadísticas mediciones de tendencia central, se espera que sus valores sean muy parecidos teniendo en cuenta las escalas propias del conjunto de datos.

2.2.2. Medidas de dispersión

En muchas aplicaciones estadísticas no basta con determinar la tendencia central de un conjunto de datos, pues puede suceder que dos variables diferentes tengan la misma medida de tendencia central, pero sus datos presenten diferente *dispersión o variabilidad*. Tener en cuenta la variabilidad de los datos y no solo la medida de tendencia central es importante en estadística porque entre menor sea la variabilidad de los datos, menor va a ser la incertidumbre en los valores estimados a partir de los mismos.

En ocasiones es importante determinar qué tan estrechos o ampliamente espaciados están los valores de una variable, de forma que la dispersión de los datos dé una idea de dónde se encuentran los altos y bajos de los datos, o si por ejemplo los valores se hallan cerca o lejos de la medida de tendencia central. Para este fin se utilizan las estadísticas de resumen denominadas *cuartiles*.

Los *cuartiles* dividen los datos en cuatro grupos, cada uno de ellos con el mismo número de valores. El primer cuartil, denominado $Q1$, hace referencia a aquel valor por debajo del cual se encuentran 25 % de los valores de la variable. De forma análoga, el tercer cuartil, denominado $Q3$, especifica aquel valor por encima del cual se encuentran 25 % de los valores, o dicho de otra forma, el valor por debajo del cual se encuentran 75 % de los valores. Siguiendo este mismo concepto, el cuartil $Q2$ corresponde a la *mediana* de la variable, de acuerdo con la definición de esta medida de tendencia central. La diferencia entre $Q1$ y $Q3$ se conoce como *rango intercuartil* o *InterQuartile Range* (IQR) y también suele reportarse como una estadística de resumen de una variable.

El concepto de cuartil se puede generalizar en estadísticas denominadas *cuantiles*, números que dividen la variable en particiones del mismo tamaño. Algunos cuantiles comúnmente utilizados son los *deciles* (10 partes) y los *percentiles* (100 partes), los

cuales son utilizados para describir el escalafón de un valor como, por ejemplo, cuando se dice que el puntaje de un estudiante en una prueba estandarizada se encuenta en el "percentil 99". Lo que quiere decir que el estudiante obtuvo un resultado igual o mejor al 99% de los otros individuos que tomaron la prueba.

Otra importante medida de variabilidad de los datos es el *rango*, el cual se define como la diferencia entre el mayor y el menor valor de una variable. Si una variable *y* cuenta con *n* valores y se asume que los valores de la variable *están ordenados ascendentemente*, el *rango* se puede expresar matemáticamente como

$$R = y_n - y_1 \tag{2.4}$$

Los principales problemas con la métrica *rango* como una medida de variabilidad tienen que ver con que es altamente dependiente de los valores extremos involucrados en su definición, así como el hecho de que solo dependa de los valores máximo y mínimo. Se desea entonces una métrica que involucre todos los valores de la variable.

Se define la *varianza muestral* de una variable *y* con *n* valores, representada convencionalmente como s^2 de la siguiente forma:

$$s^2 = \frac{\sum_{i=1}^{n}(y_i - \overline{y})^2}{n-1} \tag{2.5}$$

El término del numerador en la anterior definición se conoce como *suma de cuadrados* y es una medida que tiene en cuenta las desviaciones o *residuales* de los valores de la variable respecto a la media. Estos valores son elevados al cuadrado para solucionar problemas relacionados con el signo, si bien se puede establecer otra función matemática como el valor absoluto para este fin. Dado que la *suma de cuadrados* es una medida creciente dependiente del tamaño de la muestra, se desea independizar de alguna forma de este tamaño, razón por la cual se divide esta cantidad entre el tamaño de la muestra *n* menos uno (1).

La razón por la cual se divide entre $n - 1$ y no entre el tamaño de la muestra *n* tiene que ver con el concepto de *grados de libertad*. Se definen los grados de libertad como el tamaño de la muestra *n* menos el número de parámetros *p* estimados a partir de los datos. Dado que la media $\bar{y}$ es estimada a partir de los datos y es el único parámetro estimado involucrado en la definición de la varianza ($p = 1$), los grados de libertad son $n - p = n - 1$.

Estrechamente relacionado con la varianza está el concepto de *desviación estándar* de la muestra representada por el símbolo *s*. Esta se define como la *raíz cuadrada de la varianza* como parámetro en muchas funciones estadísticas.

- Ejemplo 5: se desea calcular el rango, la varianza muestral y la desviación estándar de los siguientes valores: 10, 12, 9, 9, 14, 13, 8, 10, 12, 12, 13, 10, 9, 14, 10.

Se reconoce en esta variable el mismo conjunto utilizado en el ejemplo anterior. Al ordenar la variable se obtiene el conjunto: 8, 9, 9, 9, 10, 10, 10, 10, 12, 12, 12, 13, 13, 14, 14. Dado que $n = 15$, $y_{15} = 14$ y $y_1 = 8$, el *rango* de la variable es

$$R = y_{15} - y_1 = 14 - 8 = 6 \tag{2.6}$$

La varianza muestral se calcula utilizando la definición y teniendo en cuenta que la media aritmética tiene un valor de 11:

$$s^2 = \frac{(10-11)^2 + (12-11)^2 + \cdots + (14-11)^2 + (10-11)^2}{14} = 3{,}857 \tag{2.7}$$

Finalmente, la desviación estándar se calcula como la raíz cuadrada de la varianza

$$s = \sqrt{3{,}857} = 1{,}963 \tag{2.8}$$

2.2.3. Correlaciones

Las correlaciones son indicadores de qué tanto dos variables se relacionan entre sí, de forma que si se conoce el valor de una variable se pueda conocer "algo" acerca del valor o comportamiento de la otra.

El rango o codominio de la función de correlación es el intervalo real [−1; 1], cuya interpretación es como sigue: una correlación de −1 entre 2 variables indica que las variables tienen una *correlación negativa perfecta*, lo cual quiere decir que un valor alto en una variable está asociado con un valor bajo en la otra variable y viceversa. Por otro lado, una correlación de 1 indica que las variables están *perfectamente correlacionadas* o tienen una correlación positiva perfecta; esto quiere decir que valores altos en una variable están asociados con valores altos en la otra. Cabe mencionar que una variable se correlaciona perfectamente con ella misma, es decir, la correlación entre una variable y ella misma siempre será igual a 1. Una correlación de 0 indica que las variables están *no correlacionadas*, lo que indica que sus valores no están asociados entre sí.

En la práctica estos valores extremos son poco frecuentes, y para interpretar correlaciones debe tenerse en cuenta qué tan cercano es el valor calculado respecto a 1, −1 o 0. A manera de ejemplo, una correlación de 0,92 entre dos variables sugiere que un incremento de cierta cantidad en una de las variables se verá reflejado en un incremento de similar cantidad en la segunda variable. De forma análoga, una correlación de −0,19 sugiere que de un incremento de cierta cantidad en una de las variables se esperará un decremento en la otra variable pero en una menor cantidad. Finalmente,

una correlación 0,00002 sugiere que las variables no están correlacionadas de forma alguna, esto es, no se esperarían cambios en una de ellas como consecuencia de movimientos en la otra.

Las medidas de correlación más comúnmente usadas son el coeficiente de correlación de Pearson, el coeficiente de correlación de Spearman y el coeficiente de Kendall.

El coeficiente de *correlación de Pearson* es una medida independiente de la escala utilizado para medir *relaciones lineales*. Se define como sigue:

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{2.9}$$

El coeficiente de correlación de Spearman (ρ) es una medida independiente de la escala y no paramétrica, esto es, no dependiente del supuesto de distribución normal de los datos. Es utilizado como indicador de la asociación tanto en relaciones lineales como no lineales. Se define reemplazando las observaciones por su rango y computando la correlación. En este contexto, el rango de una observación es el lugar ordinal que le corresponde a una observación en un escalafón o ranking como, por ejemplo, "primero", "segundo", etc.

Un tercer método de correlación es el denominado coeficiente de Kendall o τ, basado en contar el número de pares concordantes y discordantes en las observaciones. Un par de puntos (x_1, y_1) y (x_2, y_2) son concordantes si la diferencia $x_1 - x_2$ tiene el mismo signo que la diferencia $y_1 - y_2$. En caso contrario se dice que son discordantes. Para el caso particular de $x_1 = x_2$ y $y_1 = y_2$, el par no se considera ni concordante ni discordante.

El coeficiente de Kendall τ se define como sigue:

$$\tau = \frac{\#concordantes - \#discordantes}{\frac{n(n-1)}{2}} \tag{2.10}$$

Al trabajar con el coeficiente de Kendall (τ) debe tenerse en cuenta que es un método computacionalmente intensivo, por lo cual es en la práctica utilizable con un conjunto de observaciones pequeño. También debe tenerse en cuenta que si existen pares que no son concordantes ni discordantes, debe adaptarse la ecuación anterior para que los valores sigan estando en el intervalo [−1; 1]. Los detalles de esta adaptación están fuera de los alcances de este libro, pero pueden encontrarse en las referencias al final del capítulo.

Otra medida de asociación es la *covarianza*. Sin embargo, esta es una medida dependiente de la escala (el valor depende de las unidades de la variable), razón por la cual su interpretación no es tan directa. Se puede afirmar, sin embargo, que entre mayor sea la covarianza entre dos variables existe mayor asociación. En este caso, los valores positivos indican asociación positiva, mientras los valores negativos indican asociación negativa.

Una nota final en cuanto a los conceptos relacionados con correlaciones tiene que ver con que *correlación no implica necesariamente causa* en el sentido de que la medida únicamente indica *qué está* ocurriendo con las variables, mas no *por qué* esto ocurre. Cabe mencionar que esto no significa que en ciertos fenómenos la medida de correlación efectivamente implique que el comportamiento de una variable es debido a la otra como consecuencia directa.

2.3. Fundamentos de estadística inferencial

Estadística inferencial hace referencia a diversos análisis que buscan obtener información de una *población* a partir de *muestras* de datos que si el experimento está diseñado correctamente representan esta población. El objetivo entonces es obtener conclusiones de la población general basándose en las muestras, para lo cual, a diferencia de la estadística descriptiva se requieren conceptos de teoría de probabilidades y teoría estadística.

Un aspecto importante a tener en cuenta en los análisis en estadística inferencial tiene que ver con si la muestra es realmente *representativa* de la población o no, así como también es crítico para los análisis determinar si la muestra es *aleatoria*, es decir, que evita sesgo hacia algún tipo de situación o estado particular. Se considera que un valor está correctamente seleccionado de forma aleatoria si tiene exactamente la misma probabilidad de ser elegido que cualquier otro valor en el conjunto de posibles valores.

Considérese, a manera de ejemplo, del tipo de aplicaciones que se resuelven utilizando técnicas de estadística inferencial el problema de determinar la intención de voto de los electores en un proceso electoral. Dado que no es posible por razones prácticas (generalmente de índole económica) entrevistar a la totalidad del electorado (*la población*) o realizar la totalidad de las encuestas, es necesario utilizar técnicas de estadística inferencial para estimar el comportamiento del total de la población a partir de una *muestra*, esto es, un subconjunto del total del electorado, posiblemente una cantidad que desde el punto de vista práctico sea posible entrevistar o encuestar.

La noción básica en estadística inferencial es la noción de *muestra aleatoria*. Cuando se habla de si la muestra es representativa y aleatoria, en el caso del ejemplo para determinar la intención de voto, se hace referencia a que la muestra seleccionada

del total del electorado debería involucrar al menos personas de distintas regiones del país, de todos los géneros, de diferentes condiciones socioeconómicas y de varias edades, pues puede suceder que un candidato tenga preferencia en un sector demográfico particular. Si solo se entrevistaran o encuestaran las personas cuya condición demográfica favorece a un candidato particular, es claro que la estimación no va a ser fiable, pues estará *sesgada* hacia el candidato favorecido por esta condición. Se enfatiza nuevamente que en una muestra aleatoria los valores deben ser elegidos con exactamente la misma probabilidad.

Otro concepto de especial relevancia en estadística en general, y en estadística inferencial en particular, es *significancia estadística* que determina si las variaciones de algo son resultado únicamente del azar o si está realmente sucediendo algún fenómeno o alguna relación que no pueda ser explicada únicamente por el azar. En ocasiones se puede concluir que la variación no es estadísticamente significativa, lo cual no debe interpretarse como algo que no es relevante o que no es un buen resultado, ya que puede ser igualmente importante saber que no existe relación alguna o que el fenómeno es resultado únicamente del azar.

Ahora bien, surge la pregunta: ¿qué es un resultado significativo? En estadística es aquel que es *poco probable* que haya ocurrido únicamente por azar.

Sin embargo, esta afirmación lleva inmediatamente a otra pregunta más perspicaz: ¿qué significa que algo sea *poco probable*?, para resolver esta pregunta los estadísticos han acordado una *convención* en la que se establece que un evento es poco probable si ocurre *menos del 5%* de las veces. Esta es una de las razones por la cual se encuentra con tanta frecuencia los números "0,95" o "0,05" en análisis de estadística inferencial.

2.3.1. Errores estándar

Considérese el problema de determinar una medida de fiabilidad (o falta de la misma) de las estimaciones que se realizan de la población, a partir de las diferentes muestras de datos que se obtienen de la misma. Las medidas de falta de fiabilidad se denominan *errores estándar*, siendo el *error estándar de la media* una medida común definida como sigue:

$$SEM_{\overline{y}} = \sqrt{\frac{s^2}{n}} = \frac{s}{\sqrt{n}} \tag{2.11}$$

Debe notarse que en la ecuación 2.11 se utiliza la varianza muestral —o experimental— s^2, definida anteriormente. En el caso de utilizar la varianza teórica de la distribución, la anterior ecuación adquiere la forma:

$$SEM = \frac{\sigma}{\sqrt{n}} \tag{2.12}$$

Como puede observarse en las anteriores ecuaciones, el error estándar es proporcional a la varianza e inversamente proporcional al tamaño de la muestra. Esto quiere decir, intuitivamente, que entre mayor sea el tamaño de la muestra o entre menor sea la varianza de la muestra, el error estándar será menor, lo cual implica una mayor fiabilidad en la estimación. La raíz cuadrada presente en la ecuación se introduce con el objetivo de unificar dimensionalmente las unidades.

El *error estándar de la media* también puede interpretarse como una medida que describe la *variación del promedio* de una muestra de n valores aleatorios con media μ y varianza σ^2. Si se repite el experimento en varias ocasiones y se calcula el *promedio* para cada experimento, se esperaría que la *distribución de este promedio* sea más estrecha —menos dispersa— que la distribución original.

La referencia a si una estadística es "muestral/experimental" o "teórica" es una de las claves de la estadística inferencial y merece un breve comentario. Debe recordarse que la idea general en estadística inferencial es *inferir* el comportamiento de la población a partir de *muestras* obtenidas utilizando experimentos bien diseñados, es decir, aleatorios y con adecuados niveles de replicación, entre otras características.

Se asume que la variable a nivel poblacional está distribuida de acuerdo con una de las distintas distribuciones *teóricas* de probabilidad como, por ejemplo, normal, uniforme, α, β, Raleigh, t, Weibull, etc. De ahí que se utilicen *estimaciones* de las estadísticas teóricas *que no son conocidas* a partir de las estadísticas que sí se pueden calcular en la muestra de datos, que no son otra cosa que las estadísticas muestrales o experimentales. Es importante no olvidar este punto, pues al utilizar las funciones para cálculos estadísticos en los paquetes de software especializados debe tenerse claro conocimiento si la función requiere una estadística de la distribución teórica o una estadística experimental obtenida de la muestra.

Finalmente, cabe mencionar que el error estándar de la media resulta una medida utilizada en las pruebas de hipótesis, tema de la siguiente sección.

2.3.2. Pruebas de hipótesis

Se define una hipótesis como una sentencia declarativa o aserción con la capacidad de ser falsable (refutable). En el pensamiento científico, las hipótesis se formulan con el objetivo de establecer una teoría, idea o predicción, que sin embargo, debe ser comprobada experimentalmente. Esta comprobación se hace mediante pruebas que demuestren de forma inequívoca que la hipótesis es verdadera o falsa a la luz de la evidencia que la respalda o refuta.

Desde el punto de vista estadístico, se cuenta con hipótesis que se aspiran a *aceptar* o *rechazar*, esto es, establecer como probablemente verdaderas o falsas a la luz de los datos. Una de las opciones a validar es la denominada *hipótesis nula*, que es una

hipótesis falsable y establece que no está pasando nada —estadísticamente hablando— como, por ejemplo, afirmar que no existe relación alguna entre dos variables. Esta hipótesis se *rechaza*, es decir, se determina como falsa, una vez que los datos muestren que la hipótesis es poco probable, entendiendo por poco probable la definición dada anteriormente.

Afirmar que una hipótesis no es rechazada y asumirla como verdadera no se consideran lo mismo desde el punto de vista estadístico, ya que existen otros factores en juego como el tamaño de la muestra o el error en los valores que pueden llevar a que no se rechace la hipótesis. Debido a esto, es relevante para el analista de datos un buen diseño experimental con adecuados niveles de replicación para incrementar la confiabilidad de los parámetros estimados y aleatoriedad para reducir el sesgo.

En el proceso de interpretar los resultados de un análisis estadístico pueden suceder dos tipos de errores: rechazar la hipótesis nula cuando esta es verdadera o aceptar la hipótesis nula cuando esta es falsa. El primero se conoce como error *Tipo I*, mientras el segundo se denomina error *Tipo II*. Se define β como la probabilidad de aceptar la hipótesis nula cuando esta es falsa. Idealmente se desea que β tenga un valor tan pequeño como sea posible, pero sucede que entre más pequeña es la probabilidad de incurrir en un error Tipo II, más alta es la probabilidad de incurrir en un error Tipo I, luego es necesario un balance.

Las pruebas de hipótesis se basan en las *estadísticas de prueba* en las cuales se calcula un valor *p*, que es un estimado de la probabilidad de que un valor más extremo que este haya ocurrido por azar cuando la hipótesis nula es verdadera. Valores grandes de *p* indican que es poco probable que la hipótesis nula sea verdadera, luego esta se rechaza y se acepta la hipótesis alternativa.

Como cualquier aspecto teórico de la matemática, la mayoría de pruebas estadísticas están basadas en supuestos que deben cumplirse si es que las predicciones obtenidas a partir de la teoría se consideran correctas. Por ejemplo, algunas pruebas estándar asumen que los datos son obtenidos a partir de una distribución normal, lo cual se determina en la práctica como bastante realista en virtud del denominado "Teorema del Límite Central". En otras pruebas estándar para comparación de medias, como la prueba *t* de Student, se asume la *constancia de varianza*. Es decir que si las varianzas son diferentes no se deberían hacer inferencias mediante comparación de medias, pues se corre el riesgo de llegar a conclusiones erróneas. Por ello es fundamental revisar en detalle cuáles son los supuestos aplicables a cada prueba estándar antes de utilizarla en algún modelo estadístico.

A continuación se explican los conceptos más importantes detrás de las principales pruebas de hipótesis.

2.3.2.1. Prueba Shapiro-Wilk

Prueba de hipótesis que ayuda a determinar si una muestra es consistente con haber sido generada a partir de una distribución normal. La hipótesis nula de la prueba es que los datos obedecen a una distribución normal y la hipótesis alternativa es que no.

2.3.2.2. Pruebas Kolmogorov-Smirnov

Esta prueba de una muestra ayuda a determinar si esta es consistente con haber sido generada a partir de una distribución de probabilidad teórica particular. La hipótesis nula establece que la muestra es generada a partir de la distribución, mientras la hipótesis alternativa afirma que no. En este sentido, la prueba Shapiro-Wilk se puede considerar como un caso particular de la prueba Kolmogorov-Smirnov aplicada a la distribución normal.

Por su parte, la prueba Kolmogorov-Smirnov de dos muestras ayuda a determinar si estas fueron generadas a partir de la misma distribución. La hipótesis nula afirma este hecho, mientras que la hipótesis alternativa la niega.

Cabe resaltar que la prueba Kolmogorov-Smirnov aplica únicamente sobre datos continuos, pero no si algunos de los valores de la muestra están repetidos. Esto implica que, por ejemplo, la prueba Kolmogorov-Smirnov no puede ser utilizada en variables que representan conteos u ocurrencias, en cuyo caso deben aplicarse otras pruebas o recurrir a la visualización para hacerse una idea de la distribución de los datos.

2.3.2.3. Prueba t de una muestra

En esta prueba se tiene un conjunto de datos $x_1, x_2 \ldots, x_n$, los cuales se asumen como realizaciones independientes de una variable aleatoria con una distribución normal con media μ y varianza σ^2, lo cual se denota como $N(\mu, \sigma^2)$. En esta prueba se desea validar la *hipótesis nula* que establece que $\mu = \mu_0$, esto es determinar si la media de la distribución es un valor constante estipulado previamente.

Vale explicar a continuación algo del procedimiento computacional involucrado en la prueba *t*. Para ello se realizan los siguientes pasos:

1. Se estiman los parámetros teóricos μ y σ a partir de los valores empíricos $\overline{x}$ y s, respectivamente.
2. Se define una estadística $t = \dfrac{\overline{x} - \mu_0}{SEM}$
3. Se determina si este valor *t* se encuentra dentro de una región de aceptación por fuera de la cual el valor *t* tendría una probabilidad de ocurrencia igual a un nivel de significancia especificado. En caso de una distribución normal existe 95 % de probabilidad de que los datos estén en la región comprendida entre $\mu \pm 2\sigma$, y po-

siblemente sea la razón detrás de la cual el nivel de significancia frecuentemente se escoge como 5%.

Si el valor *t* se encuentra por fuera de la región de aceptación, la hipótesis nula *se rechaza*. De forma equivalente se puede calcular la probabilidad de obtener un valor mayor que el valor observado *t* y *rechazar la hipótesis* si ese valor calculado *es menor que el nivel de significancia* deseado. Como se mencionó anteriormente, esta probabilidad se denomina *p* y es el valor frecuentemente reportado en los paquetes estadísticos en las funciones que realizan las pruebas de hipótesis.

2.3.2.4. Prueba t de dos muestras

La prueba *t* de dos muestras se utiliza para probar que dos muestras provienen de distribuciones con la misma media. Sin entrar en detalles, la teoría detrás de esta prueba es la siguiente:

1. Se asume que ambas muestras son tomadas de distribuciones normales $N(\mu^1, \sigma_1^2)$ y $N(\mu^2, \sigma_2^2)$.
2. Se calcula la estadística $t = \frac{\bar{x}_2 - \bar{x}_1}{SEDM}$, en donde $SEDM = \sqrt{SEM_1^2 + SEM_2^2}$

   En este cálculo se asume que ambos grupos tienen la misma varianza.

También es posible definir un procedimiento sin tener en cuenta el supuesto de varianzas iguales. La diferencia entre ambos métodos es que en el método que tiene en cuenta las varianzas, la estadística *t* sigue una distribución *t* con un número de grados de libertad entero.

En contraparte, el método que no asume varianzas iguales a *t* no sigue una distribución *t*, pero esta distribución puede ser aproximada con una distribución *t* con la diferencia de que los grados de libertad, en general, no son un número entero. En la práctica, ambos procedimientos ofrecen un resultado similar, a no ser que tanto las desviaciones estándar como el tamaño de las muestras difieran significativamente.

2.3.2.5. Prueba F de comparación de varianzas

En ocasiones se puede estar interesado en probar el supuesto de igualdad de varianzas en lugar de darlo por hecho. Para ello se utiliza la prueba *F*, cuya hipótesis nula establece que la proporción de ambas varianzas es igual a 1, mientras la hipótesis alternativa establece que no lo son. Esta prueba asume que los datos fueron generados a partir de la distribución normal y que ambas muestras son *independientes*.

2.3.2.6. Prueba Wilcoxon de una muestra

En ocasiones es necesario no suponer que los datos provienen de una distribución normal. Para este fin, los métodos estadísticos *no paramétricos* resultan más convenientes en la mayoría de veces. La idea detrás de la prueba Wilcoxon de una muestra, es asumir que la distribución es simétrica respecto a la media teórica μ_0 y realizar una suma de las diferencias entre el valor y la media μ_0 ignorando el signo. La estadística de prueba corresponde a seleccionar un número entre 1 y el valor *n* (tamaño de la muestra) con una probabilidad de 0,5 y calcular la suma. De esta forma, se puede constatar que la distribución de la estadística de prueba puede ser calculada de forma exacta.

2.3.2.7. Prueba t de muestras dependientes

Las pruebas dependientes o apareadas son necesarias cuando las muestras *no son independientes*, lo cual puede suceder como consecuencia de experimentos con mediciones repetidas sobre los mismos sujetos experimentales o en muestras con pares de valores con similares unidades estadísticas.

2.3.2.8. Prueba de hipótesis de correlaciones

Esta prueba determina si la correlación es estadísticamente significativa. La hipótesis nula de la prueba es "no hay correlación entre las variables" (o de forma equivalente, que la correlación entre ambas variables es 0). La hipótesis alternativa afirmaría entonces que existe *algún* tipo de correlación entre las variables.

2.3.3. Modelos de regresión lineales

Los análisis de regresión son técnicas de modelado estadístico que ayudan a entender cómo una variable dependiente se comporta en función de otra(s) variable(s) independiente(s) o predictora(s). La aplicación más común de los modelos de regresión es la realización de *predicciones* o *pronósticos* basados en los datos con los que cuenta el analista.

Los modelos de regresión *lineales* son la forma más básica de los modelos de regresión. En un modelo de este tipo se plantea la posibilidad que la variable dependiente tenga un comportamiento lineal en función de las variables independientes. Si Y es la variable dependiente y X_i representa cada una de las variables independientes, el modelo lineal general se plantea mediante la ecuación de regresión:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \tag{2.13}$$

En su versión más sencilla, con una única variable independiente, el modelo lineal está dado por la expresión:

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{2.14}$$

expresión que también se encuentra de forma alternativa en la literatura como

$$Y_i = \alpha + \beta X_i + \varepsilon_i \tag{2.15}$$

Cada ε_i se asume independiente y distribuido normalmente $N(0, \sigma^2)$. La parte no aleatoria de la expresión describe la ecuación de una línea recta con α siendo el punto de interceptación del eje "y" y β representando la pendiente de la línea o *coeficiente de regresión*. Todos estos parámetros pueden ser calculados mediante el método de mínimos cuadrados. Cabe mencionar que utilizando este método, se obtienen *expresiones cerradas* de estos valores, no estimaciones de estos mismos.

El objetivo de un modelo de regresión lineal es estimar cada uno de los coeficientes βi; sin embargo, estos coeficientes son solo parte de la información que ofrece el modelo de regresión. Otra información relevante del modelo de regresión son el valor *p*, el valor R^2 y el valor R^2 ajustado.

El valor *p* es un indicador de significancia estadística. En este caso, debe notarse que tanto los coeficientes individuales estimados como el modelo en general tienen valores *p* asociados. Es recomendable contar con coeficientes y modelos estadísticamente significativos, pues este indicador descarta que los resultados hayan ocurrido únicamente como resultado aleatorio. El valor de *p* suele calcularse utilizando pruebas *t* en cada uno de los valores.

El valor R^2 mide qué tan bien el modelo lineal se adapta a los datos y, por consiguiente, qué tanta capacidad predictiva tiene el modelo. En un modelo de regresión lineal simple es igual al coeficiente de correlación de Pearson, esto es $R^2 = r^2$. Se define como el porcentaje de la varianza en la variable dependiente que puede ser explicado por el modelo de regresión. Al tratarse de un porcentaje, su valor se encuentra entre 0 y 1,0, siendo deseable valores cercanos a 1,0. Indicadores R^2 con valores bajos, señalan que el modelo lineal utilizado no es bueno para predecir el comportamiento de la variable, lo cual puede suceder por la presencia de otras variables que pueden afectar la variable dependiente y que no fueron tenidas en cuenta en el modelo. En estos casos, debe utilizarse un modelo de regresión con más variables independientes que aporten información relevante. Un modelo más complejo, como una regresión no lineal, o quizás otras técnicas de análisis de datos como aquellas basadas en aprendizaje de máquina.

El último valor R^2 ajustado, si se multiplica por 100% puede interpretarse como el porcentaje de reducción en la varianza y como tal puede ser un valor negativo.

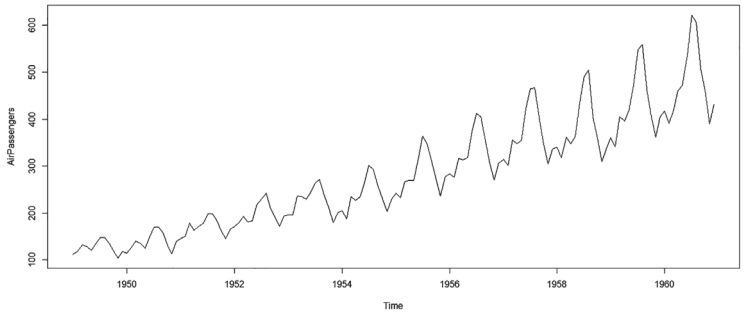
Otros valores que pueden encontrarse en los modelos de regresión hacen referencia a la distribución de los *residuales*, es decir, la diferencia entre los valores reales y los valores estimados por el modelo.

2.4. Fundamentos de series de tiempo

En ocasiones es necesario realizar la medición de una variable secuencialmente a través del tiempo como, por ejemplo, la tasa de cambio de una moneda, las temperaturas día a día de un lugar o ciudad, estadísticas demográficas o macroeconómicas anuales, ventas por día o el precio de un producto o servicio semana a semana. Cuando una variable es medida secuencialmente a través del tiempo en intervalos regulares, llamado *intervalo de muestreo*, los datos resultantes forman una *serie de tiempo*. Desde el punto de vista estadístico, *las series de tiempo* también se conocen como *procesos estocásticos de tiempo discreto*, si bien el primer nombre es más corto, suele ser el preferido.

La diferencia principal de las series de tiempo con otros modelos estadísticos es que las observaciones que están cerca la una de la otra a nivel temporal suelen estar *altamente correlacionadas* (también se utiliza el término *serialmente dependientes*). Gran parte de la metodología de los análisis de series de tiempo está basada en explicar esta correlación, utilizando métodos y modelos descriptivos y estadísticos. La figura 2.1. ilustra una serie de tiempo típica, correspondiente al número de pasajeros mensuales de una aerolínea en el período 1949-1960. Este ejemplo viene incluído en la instalación básica de la plataforma R.

Figura 2.1. Ejemplo de serie de tiempo



Fuente: Elaboración propia

Los estudios de series de tiempos deben responder dos preguntas relacionadas con: 1) *describir* la variable o 2) predecir o *pronosticar* los valores futuros de la variable. Pronosticar los futuros valores de una serie de tiempo tiene importantes aplicaciones prácticas, como:

- Predicción de mercados financieros o tendencias del valor de monedas, acciones u otros instrumentos financieros.
- Predecir demandas futuras de transporte aéreo o terrestre.
- Estudios de clima y de terremotos.
- Predecir la demanda de un producto y ventas futuras.
- Entender tendencias en el comportamiento de la población.

2.4.1. Suavizado y descomposición de series de tiempo

El primer paso para describir la serie de tiempo generalmente es recurrir a graficarla. Cuando se grafica una serie de tiempo es común encontrar varios patrones, como la *tendencia* (la variable crece, decrece o se mantiene), la *estacionalidad*, esto es, determinar si parece haber algún comportamiento periódico y determinar si el componente de error o *fluctuaciones* es significativo. En ocasiones las *fluctuaciones* son en extremo significativas e impiden determinar los patrones de tendencia o estacionalidad, por lo cual suele realizarse un proceso de "suavizado" de la serie de tiempo, es decir, un procesamiento de los datos para mitigar un poco el efecto de las fluctuaciones.

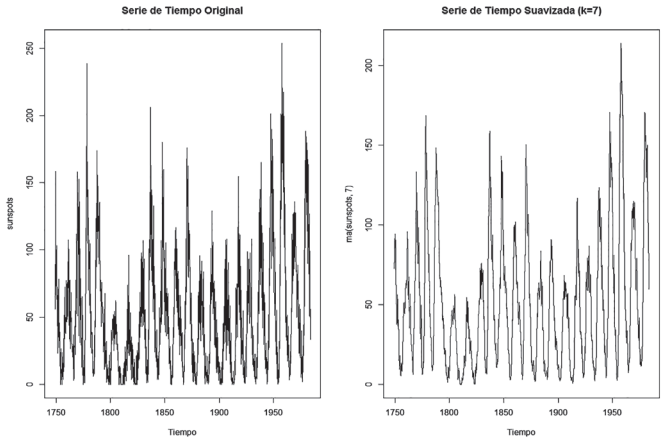
Uno de los métodos más comunes para realizar este "suavizado" es la técnica denominada *promedio móvil*. Esta consiste en reemplazar cada una de las observaciones por la *media* aritmética de esa observación y un número q de observaciones *antes* y *después* de la observación a reemplazar. Esto se expresa como:

$$S_t = \frac{Y_{t-q} + \cdots + Y_t + \cdots + Y_{t+q}}{2q+1} \tag{2.16}$$

El valor $k = 2q + 1$ es escogido para ser un valor *impar*. Debe notarse de esta formulación que la serie de tiempo *suavizada* "pierde" las primeras y últimas q observaciones.

La figura 2.2 muestra una serie de tiempo denominada `sunspots` que hace parte de las series de tiempo de ejemplo de la plataforma R que ha sido suavizada utilizando la técnica de promedio móvil con un valor $k = 7$. Debe notarse cómo la influencia de las fluctuaciones disminuye después del procesamiento.

Figura 2.2. Ejemplo de serie de tiempo duavizada. Técnica promedio móvil



Fuente: Elaboración propia

A medida que el valor de *k* aumenta, el efecto de las fluctuaciones disminuye. Surge entonces la pregunta ¿cuál debe ser el valor de *k* a utilizar en el proceso? En general no es posible determinar a priori cuál debe ser este valor, y por ello es recomendable graficar las series de tiempo procesadas para varios valores de *k*, siempre buscando que en el proceso de suavizado *no se pierdan los principales patrones encontrados* en la serie de tiempo. Cabe mencionar sin embargo que valores de $k = 5$ o $k = 7$ suelen ser valores convenientes para muchas series de tiempo encontradas en las aplicaciones.

Las series de tiempo que tienen estacionalidad pueden descomponerse en tres diferentes componentes: uno de tendencia, que captura los cambios de nivel en el tiempo; uno estacional, que captura los efectos cíclicos o periódicos debido al componente temporal y uno irregular o *error*, que captura aquellas influencias no explicadas por los otros dos componentes. Esta descomposición puede ser *aditiva* o *multiplicativa*.

El modelo aditivo establece que cada observación en un tiempo *t* es el resultado de sumar un valor debido al componente tendencia, un valor debido al componente de estacionalidad y un valor debido al componente de error o irregularidad. Matemáticamente esto se expresa como:

$$Y_t = T_t + S_t + I_t \tag{2.17}$$

De forma similar el modelo multiplicativo define que cada observación es el resultado de multiplicar los valores debido a los tres componentes.

$$Y_t = T_t * S_t * I_t \tag{2.18}$$

Algunas funciones para realizar la descomposición en estos tres factores encontrados en paquetes estadísticos como R solo están en capacidad de manejar modelos de tipo

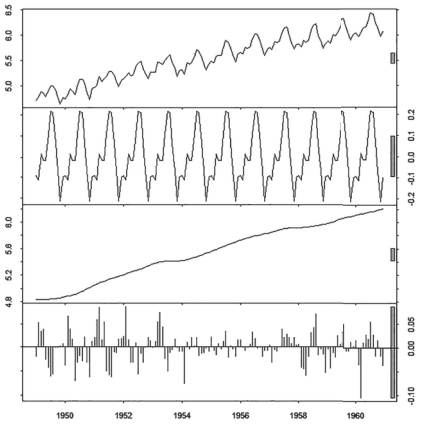
aditivo, lo cual no quiere decir que no se puedan trabajar modelos multiplicativos. Para ello debe notarse que utilizando una transformación logarítmica es posible convertir un modelo multiplicativo en uno aditivo:

$$Y_t = T_t * S_t * I_t \tag{2.19}$$

Después de tener el modelo aditivo ajustado, los resultados pueden volverse a la escala original utilizando la función de exponenciación.

La figura 2.3 ilustra un ejemplo de la descomposición de una serie de tiempo en un distintos componentes. La serie descompuesta corresponde a la originalmente mostrada en la figura 2.1 encontrada en los ejemplos de la plataforma R.

Figura 2.3. Ejemplo de descomposición de una serie de tiempo.
a) Serie original. b) Estacionalidad. c) Tendencia. d) Irregularidades



Fuente: Elaboración propia

2.4.2. Pronósticos y series de tiempo

En la sección anterior se trabajó la problemática de describir una serie de tiempo, utilizando para ello un modelo de descomposición aditivo o multiplicativo. Sin embargo, aún no se ha abordado el problema de predecir los valores futuros de la serie, es decir, *pronosticar*. Este importante aspecto es el tema de esta sección, en la que se introducirá lo relativo a *modelos exponenciales*, uno de los enfoques más comunes para el pronóstico de valores futuros de una variable.

2.4.2.1. Modelos exponenciales

Los modelos exponenciales son más sencillos que otros tipos de modelos de pronóstico, pero aún así encuentran aplicaciones en predicciones de *corto plazo*. Dependiendo

de si la serie tiene o no componentes de tendencia o estacionalidad, se utiliza un modelo exponencial particular, siendo los más comunes los siguientes.

- Un *modelo exponencial simple* se utiliza en aquellas series de tiempo que únicamente cuenta con un componente de error (irregular) alrededor de un nivel constante, esto quiere decir que la serie no tiene un componente de tendencia ni un componente de estacionalidad.
- Un modelo *exponencial suavizado de Holt* o "doble modelo exponencial" es utilizado en aquellas series de tiempo que cuentan con un componente de error alrededor de un nivel, así como un componente de tendencia.
- Un modelo *Holt-Winters* permite realizar pronósticos en series con componentes de tendencia, estacionalidad e irregularidades alrededor de un nivel o valor.

Un *modelo exponencial* simple asume que cada observación de la serie de tiempo puede ser descrita como una constante *l* y un componente de error, de forma que $Y_t = l + I_t$. La predicción del valor Y_t+1 se realiza como un promedio ponderado del valor actual y de los últimos valores de la serie según el siguiente modelo:

$$Y_{t+1} = c_0 Y_t + c_1 Y_{t-1} + c_2 Y_{t-2} + c_3 Y_{t-3} + \cdot \tag{2.20}$$

donde $c_i = \alpha(1 - \alpha)$i y $0 \leq \alpha \leq 1$. También se cumple la condición $\sum_{i=0}^{n} c_i = 1$

El valor α es un parámetro que controla qué tan rápido decaen los pesos de cada una de las observaciones pasadas. Un valor cercano a 1 pondera con mayor peso las observaciones recientes a la pronosticada, mientras un valor cercano a 0 le da más peso a las observaciones pasadas. Afortunadamente es posible establecer un criterio de optimización que permite obtener este valor α sin necesidad de recurrir a complejos procesos de calibración del parámetro. Si bien estos detalles se encuentran por fuera de los alcances del libro, basta decir que las plataformas computacionales —incluyendo R— incorporan estas técnicas de optimización del parámetro.

Un *modelo exponencial suavizado de Holt* permite ajustar series de tiempo asumiendo que la observación en el tiempo *t* puede modelarse como $Y_t = l + b * t + I_t$. La variable *b* representa la componente de tendencia como la pendiente de una recta. De forma similar al modelo exponencial simple, existe un parámetro α que controla la tasa de decrecimiento del nivel *l*, pero adicionalmente el modelo introduce un parámetro β que controla la tasa de decrecimiento de la pendiente *s*. La interpretación de estos parámetros es similar al modelo exponencial simple en el sentido de tener en cuenta las observaciones más recientes o las más antiguas.

Un *modelo Holt-Winters* permite ajustar series de tiempo que cuenten con componente de estacionalidad, adicional a la componente de nivel, tendencia y error. El

modelo asume que cada observación en un tiempo t puede ser representada como $Y_t = l + b * t + S_t + I_t$. S_t representa la influencia de la estacionalidad en un tiempo t.

Finalmente, el modelo cuenta con un nuevo parámetro γ que controla la tasa de decrecimiento de este parámetro S_t, adicionalmente a los controles α y β de los otros parámetros.

Referencias bibliográficas

[1] G. J. Myatt y W. P. Johnson, *Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining*. Second Edition. New Jersey: John Wiley and Sons Inc., 2014. [En línea]. Disponible en: http://bit.ly/30yuooJ

[2] M. J. Crawley, *Statistics: An introduction using R*. Second Edition, London: John Wiley & Sons Inc., 2015. [En línea]. Disponible en: http://bit.ly/2YUxfrF

[3] P. Dalgaard, *Introductory Statistics with R*. Second Edition, Springer, 2008. [En línea]. Disponible en: http://bit.ly/30yYw3k

[4] T. Hothorn y B. S. Everitt, *A Handbook of Statistical Analyses Using R*. 3rd Edition, New York: Chapma and Hall / CRC Press, 2014. [En línea]. Disponible en: http://bit.ly/2SiabRf

[5] D. Nolan y D. Temple Lang, *Data Science in R: a Case Studies Approach to Computational Reasoning and Problem Solving*. Boca Ratón Florida: Taylor & Francis Group / CRC Press, 2015. [En línea]. Disponible en: https://amzn.to/32umleH

[6] A. B. Downey, *Think Stats. Exploratory Data Analysis*. Second Edition, USA Highway Sebastopol, CA: O'Reilly Media Inc. 2015. [En línea]. Disponible en: https://amzn.to/2LVjh5g

[7] S. Stowell, *Using R for Statistics*. Apress, 2014. [En línea]. Disponible en: https://amzn.to/2XGIoQx

[8] J. M. Quick, *Statistical Analysis with R. Beginner's Guide*. Packt Publishing, 2010. [En línea]. Disponible en: https://amzn.to/2XM3soM

[9] A. Reinhart, *Statistics Done Wrong. The Woefully Complete Guide*. San Francisco, CA: No Starch Press, Inc., 2015. [En línea]. Disponible en: https://amzn.to/2JBh6lT

[10] S. Boslaugh y P.A. Watters, *Statistics In A Nutshell*. O'Reilly Media Inc., 2008. [En línea]. Disponible en: http://bit.ly/2Yan8Sq

[11] A. Mood, F. A. Graybill y D. Boes, *Introduction to the Theory of Statistics*. Third Edition, McGraw Hill. 1974. [En línea]. Disponible en: https://amzn.to/32v5MPK

3. Fundamentos de análisis estadístico en R

Un lenguaje de programación es de bajo nivel cuando lo irrelevante requiere demasiada atención.
—Alan J. Perlis

Este capítulo está enfocado a describir y analizar el sistema de programación estadística R, por consiguiente incluye la programación básica en lenguaje R, haciendo énfasis en la manipulación de la estructura de datos data frame; adicionalmente, muestra la forma de calcular en R las estadísticas introducidas de manera conceptual en el capítulo anterior, introduce el tema de visualización de datos en R, aprovechando las potentes capacidades de R para este fin y haciendo énfasis en la información que es posible obtener de las gráficas de las variables de un conjunto de datos. Esta información es fundamental en las primeras etapas del análisis y desempeña un rol destacado en exámenes posteriores.

3.1. Introducción

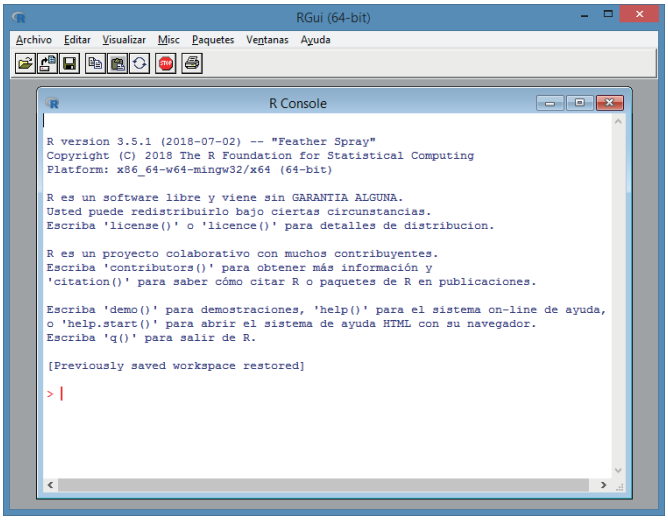
R es un lenguaje de alto nivel y entorno computacional para realizar análisis estadístico y gráficos de alta calidad. Entre sus principales características se encuentran la gran cantidad de modelos estadísticos que soporta (análisis de una variable, análisis de datos categóricos, pruebas de hipótesis, modelos lineales generalizados, análisis multivariado, series de tiempo); los miles de paquetes complementarios disponibles (adicionales a su distribución básica); el que sea software libre y de código abierto; y la posibilidad de ser ejecutado en varias plataformas, como Windows, Mac o Linux, lo que evidencia que R es uno de los lenguajes más utilizados a nivel mundial en modelos estadísticos, analítica predictiva y visualización de datos, razón por la cual ha sido escogido como herramienta tecnológica en este libro.

A diferencia de otras herramientas de análisis de datos, R es un *lenguaje interpretado* y presenta un entorno *basado en comandos*, lo cual puede dificultar el proceso de aprendizaje de la herramienta, sobre todo en usuarios acostumbrados a herramientas gráficas. Si bien esto puede verse como una desventaja, la línea de comandos permite obtener mayor *expresividad*, además de poder obtener *repetibilidad*; mediante comandos es posible escribir programas o *guiones* que permiten repetir los experimentos y

procesos, así como compartir resultados de maneras que no son posibles en otras interfaces que no están basadas en comandos.

Para minimizar la tarea de acostumbrase a las plataformas basadas en comandos, existen entornos integrados de desarrollo o *Integrated Development Environment* (IDEs), entre los cuales se encuentran RStudio, Jupyter, Eclipse, o la misma distribución estándar que se obtiene después de una instalación básica de R. El objetivo de estos entornos es facilitar el desarrollo de proyectos en R con funcionalidades similares a las encontradas al desarrollar software de otra índole, entre las que se encuentran el autocompletado de funciones, la posibilidad de utilizar control de versiones, los depuradores de código, los ejecutores paso a paso y otras. Las figuras 3.1. y 3.2. muestran cómo lucen las interfaces de la distribución estándar de R y RStudio, respectivamente.

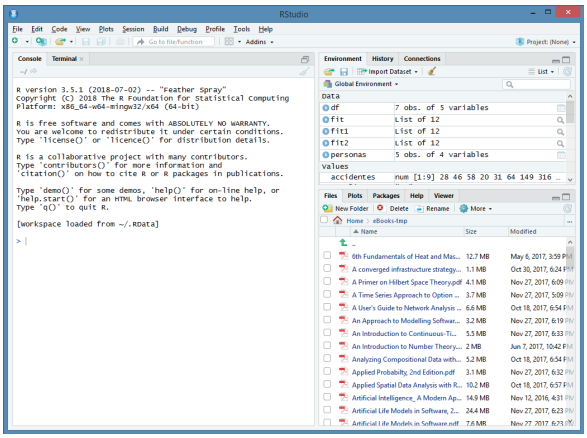
Figura 3.1. Distribución estándar de R



Fuente: Elaboración propia

El código R presentado en este libro funciona en cualquier implementación de R en su versión 3.4.x o posterior, aunque es posible que también funcione en versiones anteriores sin ninguna o con mínimas modificaciones.

Figura 3.2. Interfaz gráfica de RStudio



Fuente: Elaboración propia

3.2. Programación en R

R no es solo una plataforma para realizar análisis estadístico, sino un lenguaje de programación de alto nivel en sí mismo. Esto implica que el lenguaje provee estructuras de datos, operadores, funciones, construcciones algorítmicas como condicionales y ciclos y una amplia librería de funciones con las cuales se puede implementar cualquier problema computacional. Sin embargo, en este libro utilizaremos aspectos del lenguaje de utilidad en el proceso de análisis de datos. Las siguientes secciones describen cada uno de estos aspectos.

3.2.1. Objetos

Un objeto en R es cualquier dato almacenado en memoria al cual se le ha dado un nombre (también denominado algunas veces como *variable simbólica*). Los principales objetos en R son: objetos simples, vectores, data frames, tablas y matrices.

Para asignar un valor a un objeto o cambiarlo por uno nuevo se utiliza el *operador de asignación*, < −. Por ejemplo, si desea que un objeto "edad" tome el valor "25", debe escribirse lo siguiente en la línea de comandos de R:

```
#Definición de un objeto "edad"
edad < - 25
```

El símbolo "#" y el texto subsecuente representa un *comentario*, esto es, un texto aclaratorio que no es ejecutado por el intérprete de R, pero resulta útil para entender lo que el código realiza. En ese sentido, los comentarios están dirigidos a hacer el código R más entendible y orientado a los humanos que leen y modifican el código que a la máquina que lo ejecuta.

Los objetos simples no necesariamente tienen que tener valores numéricos. R permite la manipulación de cadenas de caracteres y cuenta con librerías (paquetes con funciones de código implementadas por terceros y disponibles para reutilización) para tratar con este tipo de dato. Para asignar un objeto simple cuyo tipo es una cadena de caracteres, se utilizan las comillas dobles (“”), tal como se muestra a continuación:

```
#Definición de cadenas de caracteres
nombre <- "José Nelson Pérez"
```

Los identificadores (nombres) de los objetos deben cumplir ciertas restricciones, relacionadas con los caracteres que deben tener (letras mayúsculas, letras minúsculas, números, puntos o guión bajo), los caracteres por los que debe comenzar (letra o punto) y que no correspondan a palabras reservadas del lenguaje. Adicionalmente, en R es diferente el identificador “EDAD” de “Edad” y “edad”, característica conocida como distinción de minúsculas/mayúsculas. Para ver el contenido de un objeto creado se ingresa en la línea de comandos su identificador.

```
edad
[1]      25
```

Un tipo de objeto muy utilizado en R es el vector, el cual guarda varios valores del mismo tipo arreglados en un orden particular, es decir, se trata de una estructura ordenada. Para crear un vector se utiliza una función de R denominada `c(...)` (el nombre viene de “combinar”) con cada uno de los componentes constituyentes del vector separados por el caracter “,”. Por ejemplo, para crear un vector con las estaturas de un grupo de personas se utiliza el siguiente comando en R:

```
#Definición de un vector
estaturas <-  c(1.84, 1.69, 1.78, 1.57, 1.92, 1.86)
```

También es posible crear vectores de otros tipos de datos como cadenas de caracteres:

```
#Definición de vector de cadenas de caracteres
nombres <-   c("Juan", "Carlos", "Mariana", "Francisco", "Pedro")
```

Cada elemento del vector tiene una posición específica que puede ser utilizada para acceder a él mediante la *notación de bracket*, la cual hace uso de “[” y “]”. Es importante dejar claro en este punto, que a diferencia de otros lenguajes de programación, en R el primer elemento del vector tiene la posición 1 y no la posición 0.

```
#Uso de la notación de bracket
estaturas [2]
[1] 1.69

nombres [1]
[1] "Juan"
```

Combinando la notación de bracket y la función `c` es posible acceder a varios elementos del vector.

```
#Acceso a varios elementos de un vector

nombres [c(2,4)]
[1] "Carlos"     "Francisco"
```

Frecuentemente se tiene la necesidad de determinar el número de componentes de un vector, esto se logra con la función length.

```
#Longitud de un vector

length (estaturas)
[2]       6

length (nombres)
[1]       5
```

Cuando se realizan análisis de datos es común encontrarse con datos faltantes, los cuales pueden resultar como consecuencia de un experimento fallido, información no disponible, errores en la captura de los datos u otras causas. Dado que la mayoría de cálculos estadísticos básicos no están pensados para lidiar con datos faltantes, los paquetes de software estadístico, y en particular R, necesitan considerar estos casos especiales y definen un valor particular para indicar este hecho. En R, este valor se denomina `NA`, y la mayoría de funciones estadísticas en R proveen la alternativa para no tener en cuenta los valores faltantes en los cálculos, ya que de otra forma el resultado del cálculo no está definido, hecho que R indica también con el valor `NA`.

```
#Valor   NA

nombres[4]       <-    NA
nombres
[1]       "Juan" "Carlos" "Mariana" NA "Pedro"
```

3.2.2. Operadores

Junto con los objetos, sean estos simples o vectores, el lenguaje R provee operadores para trabajar sobre esos objetos. Los operadores provistos por R se pueden clasificar como: aritméticos, relacionales y lógicos, estos actúan sobre elementos simples y sobre vectores; cuando operan sobre vectores las operaciones son realizadas componente a componente, siempre y cuando estos tengan la misma longitud. En caso de no tener la misma longitud, el vector con menor longitud es "reciclado", esto es, se repite tantas veces como sea necesario hasta lograr la misma longitud del otro vector. La tabla 3.1 ilustra los principales operadores aritméticos y el símbolo utilizado por R para llevar a cabo la operación.

Tabla 3.1. Operadores aritméticos en R

Operación	Símbolo
Adición	+
Sustracción	−
Multiplicación	*
División	/
División entera	%/%
Módulo	%%
Exponenciación	^

Fuente: Elaboración propia

Si una expresión está compuesta de varios operadores aritméticos, esta es evaluada en el orden de precedencia usual: exponenciación, división, multiplicación, adición y sustracción. Sin embargo, es ampliamente recomendado utilizar los paréntesis para evitar cualquier ambigüedad.

```
#Operadores aritméticos en R


(3.84*(14.62 + 64.12))/20.78
[1] 14.55061
```

Los operadores relacionales toman dos objetos de un conjunto y dan por resultado de la operación un valor lógico, valor que solo puede ser verdadero (TRUE) o falso (FALSE). Usualmente, son utilizados para comparar los objetos respecto de algún criterio establecido. La tabla 3.2 ilustra los principales operadores relacionales y el símbolo utilizado por R para llevar a cabo la operación.

Tabla 3.2. Operadores relacionales en R

Operación	Símbolo
Menor que	<
Mayor que	>
Menor o igual que	<=
Mayor o igual que	>=
Igualdad	==
Desigualdad	!=

Fuente: Elaboración propia

Finalmente, los operadores lógicos son utilizados para evaluar y definir nuevas expresiones compuestas de valores lógicos (Verdadero (TRUE) o Falso (FALSE)). Si bien es posible definir varios operadores lógicos, los comúnmente utilizados son la conjunción "y" lógico, la disyunción "o" lógico y la negación. La tabla 3.3 contiene los principales operadores lógicos y el símbolo utilizado por R para llevar a cabo la operación.

Tabla 3.3. Operadores lógicos en R

Operación	Símbolo
Conjunción	&
Disyunción	|
Negación	!

Fuente: Elaboración propia

```
#Expresión relacional

(4.47 <= 10) & (17.49 > 8.19)
[1] TRUE
```

3.2.3. Funciones

Una función es un conjunto de comandos que en conjunto realizan una tarea específica, produciendo como resultado algún tipo de salida en función de ciertos datos de entrada, aunque en verdad algunas funciones no requieren datos de entrada (también conocidos como *argumentos o parámetros*). R provee gran cantidad de funciones predefinidas, las cuales varían desde funciones para manipulación de datos numéricos y análisis estadísticos complejos hasta manipulación de archivos. También se pueden definir funciones propias, como se explicará posteriormente.

Toda función en el lenguaje R tiene un nombre y un listado de datos de entrada, aunque algunas funciones no necesitan de estos para ejecutar su funcionalidad. Los datos de entrada pueden ser constantes u otros objetos, dependiendo de la especificación de la misma. Para llamar la función se escribe el nombre de la misma y entre paréntesis "(" y ")" cada uno de los datos de entrada separados por el carácter "," respetando el orden o posición de definición de las entradas, según se muestra a continuación:

```
# Invocación de una función
nombre_funcion (entrada_1, entrada_2, ...)
```

Dependiendo de cómo haya sido definida la función, es posible obviar alguno(s) de los datos de la entrada, ya que se les pudo haber asignado un valor por defecto como parte de su definición. También es posible alterar el orden de las entradas a la función, colocando en el llamado el nombre del parámetro (también conocido como "argumento identificado", cuyo nombre es diferente al nombre de la función) y el valor a asignar a esa entrada junto con el operador de asignación "=". Como resultado de invocar o llamar la función, se produce un resultado que puede ser almacenado en un objeto.

A manera de ejemplo, considérese la función `round` encontrada en la librería de R, diseñada para aproximar un número a su valor más cercano dependiendo de un número de dígitos de precisión. Una primera versión de un llamado de esta función únicamente recibe como entrada el número, ya que por defecto el número de dígitos decimales utilizados en la aproximación es cero.

```
# Invocación de una función
# con argumentos por defecto
round (3.141592)
  [1] 3
```

Para aproximar a un número diferente de dígitos, se utiliza la misma función `round`, pero se invoca con diferentes parámetros de entrada. Para hacer esto se utiliza una segunda versión del llamado, utilizando el parámetro de entrada `digits` con un valor igual a 2 para indicar que la aproximación debe realizarse con dos dígitos de precisión.

```
# Invocación de una función
# con argumentos identificados

Round (3.141592, digits=2)
  [1] 3.14
```

En el ejemplo anterior se observa que hay parámetros, como el número a redondear, que son obligatorios, mientras que otros son opcionales. ¿Cómo saber esa información?, ¿cómo se supo que el nombre del segundo parámetro era "digits"? Para resolver estas preguntas, R provee un detallado archivo de ayuda para cada una de sus funciones, ayuda que puede ser directamente accesible desde el entorno de desarrollo mediante la función "`help`" o anteponiendo al nombre de la función el carácter interrogante "`?`". También es posible utilizar la función `args` para ver los argumentos de entrada a una función, únicamente sin la descripción y uso de la misma.

```
# Invocación de archivo de ayuda
help(nombre_funcion)
# Invocación de archivo de ayuda
#   Método alternativo
?nombre_funcion
# Ayuda:   argumentos de entrada de una función
args (nombre_funcion)
```

Debido a que el principal campo de aplicación de R es el análisis estadístico, es común que junto con el gran número de funciones provistas por R se cuenten con conjuntos de datos sobre los cuales realizar estos análisis. Las funciones y los conjuntos de datos son organizados en *paquetes*, algunos de los cuales deben ser cargados en la memoria principal del entorno para que sus funciones y conjuntos de datos estén disponibles en la sesión.

El directorio donde residen los diferentes paquetes que se han descargado en el computador que ejecuta el sistema R se denomina la librería. La función `library()` lista los paquetes salvados en el computador, mientras la función `sessionInfo()` muestra todos aquellos paquetes que además de estar instalados en el computador están ya cargados en la sesión R. Para obtener los distintos conjuntos de datos disponibles en la sesión, se invoca la función `data()`. Tanto la función `library()` como la función `data()` abren una ventana adicional con la información consultada en lugar de imprimir la respuesta al comando inmediatamente después de ejecutarlo.

```
#Listar paquetes e información de la sesión
library()
sessionInfo()R version 3.4.2 (2017-09-28)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 8.1 x64 (build 9600)
Matrix products: default
locale:
[1] LC_COLLATE=Spanish_Colombia.1252 LC_CTYPE=Spanish_Colombia.1252
```

```
[3] LC_MONETARY=Spanish_Colombia.1252 LC_NUMERIC=C
[5] LC_TIME=Spanish_Colombia.1252
attached base packages:
[1] stats   graphics   grDevices  utils    datasets   methods   base
loaded via a namespace   (and not attached):
[1] compiler_3.4.2
data ()
```

Dentro de las funciones más populares encontradas en R, es común hallar las de tipo matemático/estadístico. La tabla 3.4 muestra algunas funciones matemáticas comunes que provee R como parte de su entorno básico. Puede hacerse uso de los archivos de ayuda para precisar los detalles de cada una de ellas. Por su parte, algunas funciones estadísticas relevantes se explicarán posteriormente.

Tabla 3.4. Funciones matemáticas comunes en R

Propósito	Función R
Exponencial	`exp`
Logaritmo natural	`log`
Logaritmo base 10	`log10`
Raíz cuadrada	`sqrt`
Coseno	`cos`
Seno	`sin`
Tangente	`tan`
Valor absoluto	`abs`
Factorial	`factorial`

Fuente: Elaboración propia

Es importante mencionar que si un vector es utilizado como argumento de entrada a una función que espera un objeto simple (como las funciones matemáticas de la tabla 3.4), R ejecuta la función por cada componente y en lugar de arrojar por resultado un único número, retorna un nuevo vector. Adicionalmente, algunas funciones están pensadas específicamente para recibir por parámetro un vector y calcular un único valor de salida utilizando todos o algunos componentes del vector en el cálculo. Ejemplos de estas son las funciones estadísticas que se utilizarán a lo largo del texto como `mean`, `max`, `sd` y otras tantas.

```
# Funciones aritméticas en R
Sqrt ( c(1.25, 4.8, 2, 20.96) )
[1]  1.118034   2.190890   1.414214   4.578209
```

Otras funciones de interés en el análisis de datos son aquellas utilizadas para crear una secuencia de valores. En particular, la función `seq(inicial,final,salto)` es utilizada para generar un vector de números desde un valor inicial a un valor final cada cierto salto. Esta función es muy utilizada en la definición de la escala de algunos gráficos. Por ejemplo, `seq(1,20,2)` genera un nuevo vector con los números del 1 al 20 en saltos de dos en dos.

```
# Creación de una secuencia
seq(1,20,2)
[1]  1   3   5   7   9   11   13   15   17   19
```

3.2.4. Factores

Es muy común en el análisis de datos contar con datos categóricos o nominales. Para definir explícitamente en R un objeto categórico se utiliza la función `factor`, en la cual se especifican los diferentes niveles o categorías como un vector.

```
#   Creación de datos categóricos
tipos.quemadura   <- factor( c("Primer Grado",
                          "Segundo Grado","Tercer Grado"))
tipos.quemadura
[1] Primer Grado    Segundo Grado    Tercer Grado
Levels:   Primer Grado    Segundo Grado    Tercer Grado
```

Si la variable categórica es además una variable ordinal, es posible indicar de forma explícita este hecho mediante los argumentos `levels` y `ordered`. En el argumento `levels` se indican todos los niveles ordenados ascendentemente de menor a mayor, mientras al argumento `ordered` se le asigna un valor `TRUE` para indicar que la variable es ordinal.

```
# Variables ordinales en R
quemaduras <-   factor(   tipos.quemadura,
                          levels = tipos.quemadura,
                          ordered=TRUE )
quemaduras
[1]  Primer Grado   Segundo Grado   Tercer Grado
Levels:  Primer Grado  <  Segundo Grado  <  Tercer Grado
```

3.2.5. Listas

En el lenguaje R, una lista es un compuesto de varios componentes que se agrupan para ser manipulados como un único objeto. En una lista en R los componentes no tienen que ser del mismo tipo, y en caso de serlo, no tienen que tener el mismo tipo

de dato ni la misma longitud (en el caso que todos los componentes sean vectores). Las listas encuentran aplicación, entre otros casos, cuando una función debe retornar más de un objeto.

Para crear una lista en R se utiliza la función `list(...)` que recibe como parámetro los distintos objetos que la componen. Cuando se crea la lista con la función `list`, si bien no es obligatorio, es de utilidad que los argumentos de la función sean identificados; si estos son argumentos nombrados o identificados, es posible hacer referencia a cada componente con una notación conocida como la "notación del signo $", en el que cada componente de la lista se identifica como `nombre_lista$nombre_argumento`. El siguiente ejemplo ilustra el proceso de creación de una lista en R. El primer paso es definir los componentes individuales que van a conformar la lista.

```
memoria_gb <-   16
num_procesadores <-   8
sistema_operativo <-   "Linux Ubuntu Server 16.04.2"
versiones_kernel <-   c("4.4.0.66", "4.4.0.72")
```

Acto seguido se define la lista haciendo uso de la función `list` y utilizando argumentos identificados en la creación.

```
# Creación de una lista
servidor1 <- list(mem=memoria_gb, proc=num_procesadores,
  so=sistema_operativo, kernels= versiones_kernel)
```

Como se utilizaron argumentos identificados, es posible utilizar la notación "del signo $" para hacer referencia a cada componente de forma individual.

```
# Acceso a componentes de una lista
servidor1$mem
[1] 16
servidor1$kernels
[1] "4.4.0.66" "4.4.0.72"
```

Si no se recurre a argumentos identificados en la creación de la lista, se debe utilizar la notación de "doble bracket" para acceder a cada uno de los componentes individuales. Cada componente es accesible mediante un número entero que representa su posición en la lista.

```
# Notación de doble bracket
servidor2 <- list(memoria_gb, num_procesadores,
  sistema_operativo, versiones_kernel)
servidor2
[ [1] ]
[1]   16
[ [2] ]
[1]   8
[ [3] ]
[1]   "Linux Ubuntu Server 16.04.2"
[ [4] ]
[1]   "4.4.0.66" "4.4.0.72"
servidor2[ [3] ]
[1]   "Linux Ubuntu Server 16.04.2"
```

La notación de "doble bracket" también aplica para las listas creadas con argumentos identificados. Sin embargo, es más común encontrar la notación del signo $ y por ello será la que se utilice a lo largo del libro.

```
# Notación de doble bracket
servidor1[ ["mem"] ]
[1]   16
servidor1[ ["kernels"] ]
[1]   "4.4.0.66" "4.4.0.72"
```

3.2.6. Manipulación de data frames

Un data frame es un tipo especial de lista en el que todos los componentes son vectores de la misma longitud, relacionados en cuanto los elementos en la misma posición correspondan a datos de la misma unidad experimental o en la terminología empleada en el libro de la misma observación. De esta forma, este objeto R corresponde con la definición encontrada en la tabla 2.1.

Para crear un data frame se utiliza la función `data.frame(...)` con cada variable que conforma el objeto como parámetros, los cuales son probables que ya estén definidos como vectores R. A manera de ejemplo, considérense los datos de varias personas (observaciones) para un estudio clínico. De cada uno de ellos se tiene información de su estatura, peso y género. La definición de esta tabla de datos se muestra a continuación:

```
# Creación de un data frame
estatura <-   c(1.68, 1.84, 1.78, 1.55, 1.89 )
peso <-   c(55, 91, 88, 47, 105)
genero <-   c("F", "M", "M", "F", "M")
personas <-   data.frame(estatura,peso,genero)
personas
   estatura      peso    genero
1 1.68           55       F
2 1.84           91       M
3 1.78           88       M
4 1.55           47       F
5 1.89           105      M
```

En ocasiones, por ejemplo, dentro de la definición de una función es necesario crear un data frame sin ningún contenido; para lograr esto debe invocarse la función `data.frame` sin ningún parámetro.

```
# Data frame sin contenido
df.vacio   <-   data.frame()
```

Posterior a la creación de la tabla de datos, es posible acceder a cada variable utilizando la notación "$", como en el caso de las listas o notación de indexación en "brackets". La notación en "bracket" puede verse como un sistema coordenado en filas y columnas, donde puede accederse a cada dato particular, a una observación (fila) completa, a una variable o incluso puede realizarse filtrado o búsquedas particulares de acuerdo con una expresión relacional.

Para acceder a un dato particular de la observación i-ésima en la variable j-ésima se utiliza la notación en "bracket" `[i,j]`, mientras que para acceder a toda la observación se utiliza la notación `[i,]`. Ejemplos de estas notaciones se ilustran a continuación utilizando la tabla de datos `personas` anteriormente creada.

```
# Indexación en data frames
personas[3, 3]
[1]   M
Levels: F  M
personas[3, 1]
[1]   1.78
personas[1, ]
  estatura       peso    genero
1 1.68           55      F
personas[5, ]
  estatura       peso    genero
5 1.89           105     M
```

Para ver los valores de una variable se pueden utilizar, aparte de la notación con signo $, la notación `[,j]`, la cual devuelve un vector o la notación `[j]`, que finalmente arroja una nueva tabla de datos.

```
# Valores de una variable
personas[, 2]
[1]     55   91   88   47   105
personas[2]
  peso
1 55
2 91
3 88
4 47
5 105
```

También es posible excluir partes del conjunto de datos en lugar de seleccionarlos con el símbolo – o escoger un intervalo de filas o columnas del conjunto de datos con la notación de intervalo: por ejemplo, si se deseara seleccionar las observaciones 2 a la 4 de la tabla de datos sin la tercera variable (la variable "genero") el código R sería:

```
# Indexación por intervalos
personas[2: 4, -3]
  estatura        peso
2 1.84            91
3 1.78            88
4 1.55            47
```

Hasta ahora se ha utilizado un enfoque basado en índices para seleccionar los datos; sin embargo, en la práctica suele ser necesario extraer aquellos datos que satisfagan una condición determinada, por ejemplo "las personas que midan más de 1,80" o "las personas de género femenino". Para realizar esto se sigue utilizando la notación "bracket", solo que en lugar de los índices a seleccionar se emplea una expresión condicional.

```
# Indexación con expresiones condicionales
personas[personas$estatura > 1.80,]
          estatura        peso    genero
2         1.84            91      M
5         1.89            105     M

# Indexación con expresiones condicionales
personas[personas$genero   ==   "F", ]
  estatura        peso     genero
1 1.68            55      F
4 1.55            47      F
```

Otra forma alternativa de obtener datos que satisfagan una condición determinada es utilizando la función `subset`. Esta es una versátil función que se puede utilizar de varias formas dependiendo de si se desea conservar los datos a nivel de observaciones (filas) o a nivel de variables (columnas). A nivel de las observaciones, la función recibe como argumento un data frame y una expresión relacional y devuelve una tabla de datos con aquellas observaciones que satisfagan el criterio de la expresión relacional; a nivel de variables, la función recibe como argumento un data frame y un argumento identificado como `select`, que toma como valor un vector con aquellas variables que se desean conservar. Esta forma de utilización también es útil para reordenar las variables en el conjunto de datos. Ambos casos se muestran en el siguiente código de ejemplo.

```
#   Función subset a nivel de fila
mujeres <- subset(personas, personas$genero == "F")
mujeres
          estatura        peso    genero
1         1.68            55      F
4         1.55            47      F
# Función subset a nivel de columna
personas1 <- subset(personas,select=c(peso,estatura))
personas1
          peso    estatura
1         55      1.68
2         91      1.84
3         88      1.78
4         47      1.55
5         105     1.89
```

La notación con el símbolo "$" también permite crear nuevas variables en una tabla de datos. Por ejemplo, se desea obtener una nueva variable en el data frame `personas` que calcule el índice de masa corporal (IMC), el cual se define como $IMC = \frac{peso}{estatura^2}$

El código R para lograr este objetivo es el siguiente:

```
# Creación de nuevas columnas
personas$IMC   <- personas$peso/personas$estatura^2
personas
  estatura        peso    genero  IMC
1         1.68    55      F       19.48696
2         1.84    91      M       26.87854
3         1.78    88      M       27.77427
4         1.55    47      F       19.56296
5         1.89    105     M       29.39447
```

Debe notarse que el cálculo de una variable realiza los cálculos simultáneamente en todas las observaciones.

Una forma alternativa de lograr lo anterior, sin modificar el conjunto de datos original, es a través de la función `transform`. Esta función recibe como parámetro una tabla de datos y como segundo parámetro el nombre de la nueva variable y la expresión para calcularla, según se muestra a continuación:

```
# Función transform
personas1 <- transform( personas,
  IMC=personas$peso/personas$estatura^2 )
personas1
  estatura       peso    genero  IMC
1        1.68    55      F       19.48696
2        1.84    91      M       26.87854
3        1.78    88      M       27.77427
4        1.55    47      F       19.56296
5        1.89    105     M       29.39447
```

3.2.7. Tablas de frecuencia

Las tablas de frecuencia son estructuras de datos que llevan un conteo del número de observaciones clasificadas en un determinado nivel de una, dos o en general n variables categóricas. El número n de variables categóricas se denomina la dimensión de la tabla.

Supóngase que se tiene un conjunto de personas e información de dos diferentes variables categóricas en cada observación: la primera variable hace referencia al color del cabello y cuenta con los niveles `{Negro, Café, Rojo, Rubio}`; por su parte, la segunda variable tiene la información del color de los ojos de la persona y cuenta con los niveles `{Café, Azul, Miel, Verde}`. Dadas estas variables, una tabla de frecuencia de dos dimensiones que lleve el conteo de las observaciones en cada combinación de niveles podría lucir como en la tabla 3.5.

Tabla 3.5. Ejemplo de "Tabla de frecuencia"

	Café	Azul	Miel	Verde
Negro	68	20	15	5
Café	119	84	54	29
Rojo	26	17	14	14
Rubio	7	94	10	16

Fuente: Elaboración propia

Lo que la tabla 3.5 refleja, es que del total de personas, existen 94 que cumplen la condición de que su color de cabello sea clasificado como "Rubio" y adicionalmente su color de ojos esté clasificado en la categoría "Azul". La interpretación es análoga para el resto de valores. A nivel de terminología, vale la pena mencionar que las tablas de frecuencia con más de un factor también se conocen como *tablas de contingencia* o *tabulaciones cruzadas* (*cross-tabulation*).

Considérese ahora el mismo ejemplo anterior, pero se cuenta con una nueva variable categórica con el género de la persona. Esta variable tiene dos niveles `{Masculino, Femenino}`. Con esta nueva adición, se puede crear una tabla de frecuencia de tres dimensiones que indique, por ejemplo, el total de personas cuyo color de cabello es "Café", su color de ojos es "Verde" y su género es "Femenino". Si bien es posible visualizar esta tabla de frecuencia en tres dimensiones, es claro que a medida que el número de dimensiones aumenta no es posible obtener una visualización conveniente. Debido a ello, es una práctica común visualizar únicamente tablas de frecuencia de dos dimensiones separadas explícitamente por cada dimensión adicional.

A manera de ejemplo, la tabla de frecuencia de tres dimensiones se visualizaría como dos tablas de frecuencias de dos dimensiones de idéntica estructura y diferentes valores, una para cada género.

Tabla 3.6. Ejemplo de "Tabla de frecuencia. Género=Masculino"

	Café	Azul	Miel	Verde
Negro	32	11	10	3
Café	53	50	25	15
Rojo	10	10	7	7
Rubio	3	30	5	8

Fuente: Elaboración propia

Tabla 3.7. Ejemplo de "Tabla de Frecuencia. Género=Femenino"

	Café	Azul	Miel	Verde
Negro	36	9	5	2
Café	66	34	29	14
Rojo	16	7	7	7
Rubio	4	64	5	8

Fuente: Elaboración propia

Para introducir las diferentes funciones proporcionadas por R para la manipulación de tablas de frecuencias, considérese una tabla de datos cuyos registros tienen la estructura mostrada en la figura 3.8. A este conjunto de datos se le ha dado el identificador `ColorCabelloOjos`.

Tabla 3.8. Conjunto de datos "ColorCabelloOjos"

Color Cabello	Color Ojos	Genero
⋮	⋮	⋮
Negro	Azul	Masculino
Café	Miel	Masculino
Rojo	Verde	Femenino
Rubio	Azul	Masculino
Café	Café	Femenino
Negro	Café	Masculino
⋮	⋮	⋮

Fuente: Elaboración propia

Dado el conjunto de datos posiblemente almacenado en un data frame, la función `table` permite crear una tabla de frecuencia. La función `table` tiene como parámetro una, dos o en general las n variables categóricas que conforman la tabla de contingencia. A manera de restricción, para que la función `table` funcione de manera adecuada, el tipo de las variables involucradas en la creación de la tabla debe ser `factor`.

```
# Creación de tablas de frecuencia
class (ColorCabelloOjos$color.cabello)
[1]   "factor"
tabla.cabello <- table ( ColorCabelloOjos$color.cabello )
tabla.cabelloojos <- table( ColorCabelloOjos$color.cabello,
                                ColorCabelloOjos$color.ojos )
tabla.cabelloojosgenero <- table( ColorCabelloOjos$color.cabello,
                                        ColorCabelloOjos$color.ojos,
                                        ColorCabelloOjos$genero )
tabla.cabelloojosgenero
, , =       Femenino
         Azul     Café   Miel    Verde
Café     34       66     29      14
Negro    9        36     5       2
Rojo     7        16     7       7
```

```
Rubio     64      4       5       8

, ,       = Masculino
          Azul    Café    Miel    Verde
Café      50      53      25      15
Negro     11      32      10      3
Rojo      10      10      7       7
Rubio     30      3       5       8
```

Las funciones `dim` y `dimnames` son las encargadas de obtener las dimensiones de la tabla de contingencia y las etiquetas o nombres de cada dimensión, respectivamente.

```
# Dimensiones de una tabla de frecuencia
dim(tabla.cabelloojosgenero)
[1]  4   4   2
dimnames(tabla.cabelloojosgenero)
[ [1] ]
[1]   "Café"   "Negro"   "Rojo"   "Rubio"
[ [2] ]
[1]   "Azul"   "Café"   "Miel"   "Verde"
[ [3] ]
[1]   "Femenino"   "Masculino"
```

También es posible indexar las tablas de frecuencia por un nivel particular de cada factor utilizando la notación de "bracket". Si no se especifica índice en una dimensión determinada se deja el índice vacío.

```
tabla.cabelloojosgenero[ ,"Miel", "Masculino" ]
Café      Negro   Rojo    Rubio
25        10      7       5

tabla.cabelloojosgenero[ "Rubio", , "Femenino" ]
Azul      Café    Miel    Verde
64        4       5       8
```

Finalmente, en ocasiones puede ser necesario convertir una tabla de contingencia en un data frame. Para este propósito, se utiliza la función `as.data.frame.table`, que recibe por parámetro la tabla de contingencia y retorna un objeto de tipo data frame. Esta tabla de datos contiene una nueva variable denominada "Freq", con la información del conteo de cada combinación encontrada en la tabla de contingencia.

```
# Conversión de tabla de frecuencia a data frame
df <- as.data.frame.table ( tabla.cabelloojosgenero )
class( df )
[1] "data.frame"
```

3.2.8. Manipulación de fechas

El entorno R puede lograr la manipulación de fechas, bien sea a través de funciones que vienen incorporadas a la plataforma por defecto o utilizando librerías especializadas. Si bien lograr manipular correctamente las variables de tipo fecha es relevante en el análisis de series de tiempo, es común encontrar gran cantidad de conjuntos de datos con campos de este tipo, por lo cual su manipulación correcta es una habilidad deseable en cualquier analista de datos.

En lo referente a la manipulación de fechas, usualmente se comienza con la función `as.Date`, función predefinida en el entorno R, es decir, disponible sin necesidad de utilizar alguna librería adicional. La función `as.Date` en su versión más sencilla recibe una cadena de caracteres y las interpreta como una fecha en formato *yyyy-mm-dd* (año expresado en cuatro dígitos, mes en dos dígitos y día). Esta versión también puede interpretar la fecha en formato *yyyy/mm/dd*.

```
# Interpretación de fechas
# Formatos por defecto
as.Date("2017-08-17")
[1]    "2017-08-17"
as.Date("2017/08/17")
[1]     "2017-08-17"
```

Debido a los distintos formatos en que se puede representar una fecha, la función `as.Date` cuenta con un parámetro `format` en el cual se especifica el formato en el que debe ser interpretada la fecha. El listado completo de los formatos se encuentra en la documentación de la función `strptime`. Los siguientes ejemplos únicamente utilizarán un subconjunto de todos los formatos disponibles.

```
# Interpretación de fechas
# Otros formatos
as.Date("21/APR/17",format="%d/%b/%y")
[1]    "2017-04-21"
as.Date("April-17-17", format="%B-%d-%y")
[1]    "2017-04-17"
as.Date("August-17-2017", format="%B-%d-%Y")
[1]    "2017-08-17"
as.Date("08-17-2017", format="%m-%d-%Y")
[1] "2017-08-17"
```

En aquellos entornos R en los que no está configurado correctamente el reloj es probable que las anteriores funciones den por resultado `NA`. Para corregir esto, debe configurarse correctamente la localización de zona horaria mediante el siguiente comando:

```
# Configuración zona horaria
Sys.setlocale("LC_TIME",   "C")
```

En aquellos formatos de fecha que involucren no solo la fecha calendario, sino la información horaria, es necesario utilizar la función `strptime`. Esta función recibe por parámetro una cadena de caracteres que representa la fecha con información horaria y un formato en el cual debe interpretarse la fecha. Dentro de este formato, los caracteres especiales `%H`, `%M`, `%S` representan horas, minutos y segundos, respectivamente. Algunos ejemplos se muestran a continuación, resaltando una vez más que la documentación de la función `strptime` contiene todo lo referente a los formatos a utilizar.

```
# Formatos de fecha y hora
strptime("21/08/2017 14:18","%d/%m/%Y %H:%M")
[1]   "2017-08-21 14:18:00 -05"
strptime("17-08-21 14:18:12","%y-%m-%d %H:%M:%S")
[1]   "2017-08-21 14:18:12 -05"
strptime("2017-Apr-21 10:05","%Y-%B-%d %H:%M")
[1]   "2017-04-21 10:05:00 -05"
```

Cuando se crea un objeto con la función `strptime` el tipo de dato asociado al objeto R es un tipo de dato conocido como `POSIXlt`. Sobre este tipo de dato es posible utilizar varias funciones para manipular las fechas, como la función `difftime`, que permite calcular el intervalo de tiempo entre dos fechas, expresado en unidades como horas, segundos, minutos, días (valor por defecto) o semanas. Las unidades son pasadas a la función `difftime` en el parámetro `units`. Otra función aplicable sobre este tipo de dato es `weekdays`, que obtiene el día de la semana correspondiente a una fecha determinada.

```
# Funciones difftime y weekdays
fecha1 <- strptime("2017-Apr-21 10:05","%Y-%B-%d  %H:%M")
fecha2 <- strptime("2017-08-30 14:18","%Y-%m-%d  %H:%M")
difftime( fecha2, fecha1, units="days" )
Time difference of 131.1757 days
difftime( fecha2, fecha1, units="hours" )
Time difference of 3148.217 hours
weekdays( fecha1 )
[1] "Friday"
```

Entre las librerías especializadas para manipulación de fechas se encuentra `lubridate`. Haciendo uso de esta es posible obtener resultados similares, con la ventaja de que no hay que especificar el formato, pues las diversas funciones encontradas en la librería están en capacidad de manipular fechas en formatos heterogéneos e interpretarlos convenientemente. Dentro de estas funciones se encuentra `mdy`, `dmy` o `ymd`, que reciben fechas en una amplia variedad de formatos y convierten a fechas en el formato estándar *yyyy-mm-dd UTC*, dependiendo de si especifica primero el año (función `ymd`), el día o el mes (como en la función `mdy`). Algunos ejemplos de cómo se utilizan se muestran a continuación.

```
# Funciones de la librería lubridate
ymd("2017/08/21")
[1]   "2017-08-21"
ymd("2017 August 21")
[1]   "2017-08-21"
dmy("21 Aug 2017")
[1]   "2017-08-21"
mdy(08212017)
[1]   "2017-08-21"
```

En ocasiones es necesario acceder a los componentes individuales de una fecha u hora, como el mes o la hora de un objeto de tipo fecha e incluso funciones algorítmicas más complejas, como determinar el día de la semana de una fecha. Afortunadamente la librería `lubridate` cuenta con todas estas funciones y otras más ya implementadas, las cuales se resumen en la tabla 3.9.

Tabla 3.9. Funciones de la librería "lubridate"

Propósito	Función R
Obtener el año de una fecha	`year`
Obtener el mes de una fecha	`month`
Obtener el número de semana de una fecha	`week`
Obtener el día de una fecha	`day`
Obtener el día del año de una fecha	`yday`
Obtener del día de la semana de una fecha	`wday`
Obtener la hora de una fecha con hora	`hour`
Obtener los minutos de una fecha con hora	`minute`
Obtener los segundos fecha con hora	`second`

Fuente: Elaboración propia

```
# Otras funciones de la librería lubridate
fecha1 <- mdy(08212017)
year( fecha1 )
[1] 2017
day( fecha1 )
[1] 21
wday( fecha1 )
[1] 2
yday( fecha1 )
[1] 233
```

3.3. Funciones estadísticas en R

Una vez se ha introducido el lenguaje R y adquirido destreza en la manipulación de sus diferentes objetos, el siguiente paso es realizar análisis estadísticos, tanto descriptivos como inferenciales, sobre los datos utilizando esta tecnología. Esta sección muestra la forma como se realizan en R los análisis estadísticos básicos introducidos de manera conceptual en las secciones 2.2. y 2.3.

3.3.1. Tipos de variable

El tipo de una variable determina cómo será utilizada por R en las gráficas y análisis estadísticos. Para determinar el tipo de dato de una variable, R provee la función `class`, que recibe como parámetro una variable de una tabla de datos y devuelve su tipo. Si bien existen múltiples tipos de variable en R, los de uso más frecuente son `numeric`,`integer`,`factor`,`character` y `POSIX1t`.

- Las variables de tipo numeric contienen números reales positivos o negativos, así como el símbolo `NA`. Es el tipo esperado para variables continuas.
- Las variables de tipo `integer` contienen números enteros positivos o negativos de forma similar al tipo `numeric`. Es el tipo esperado para conteos.
- Las variables de tipo `factor` contienen valores únicos (niveles), los cuales pueden ser número o caracteres. Es el tipo esperado para variables categóricas.
- El tipo `character` contiene cadenas de caracteres utilizados para almacenar información no numérica.
- El tipo de dato `POSIX1t` se explicó en una sección anterior referente a la manipulación de fechas.

```
# Tipo de variable
class(personas$estatura)
[1]   "numeric"
class(personas$genero)
[1]   "factor"
```

Es posible que el tipo de una variable en una tabla de datos no resulte apropiado para su análisis. Esto puede suceder porque en ocasiones, al importar de un archivo, R automáticamente asigna el tipo basado en el contenido de la variable. Por ello, R provee funciones que permiten convertir el tipo de variable a otro, las cuales se encuentran resumidas en la tabla 3.10.

Tabla 3.10. Funciones de conversión de tipo de variable en R

Propósito	Función R
Convertir a `numeric`	`as.numeric`
Convertir a `integer`	`as.integer`
Convertir a `factor`	`as.factor`
Convertir a `character`	`as.character`

Fuente: Elaboración propia

3.3.2. Estadísticas de resumen en R

La tabla 3.11 muestra las funciones R utilizadas para calcular las estadísticas de resumen de una variable en un conjunto de datos. Cada una de ellas recibe como parámetro un vector o una variable de una tabla de datos, a excepción de la función `summary`, que puede ser directamente aplicada a la tabla de datos y calcula las estadísticas de resumen de todas las variables presentes en la misma.

Tabla 3.11. Funciones estadísticas comunes en R

Propósito	Función R
Media aritmética	`mean`
Mediana	`median`
Desviación estándar	`sd`
Rango de la variable	`range`
Cuantiles experimentales	`quantile`
Máximo	`max`
Mínimo	`min`
Resumen	`summary`

Fuente: Elaboración propia

Un tipo de análisis muy común se presenta cuando es necesario calcular una estadística de resumen de una variable continua para cada nivel de una o más variables categóricas. Por ejemplo, se desea saber la estatura y el peso promedio (variables continuas) de todas las mujeres y hombres presentes en el estudio, teniendo en cuenta que es una variable categórica con dos niveles "F" y "M". La forma de lograr esto es haciendo uso de la función `aggregate`, que permite calcular estadísticas "de grupo", esto es, por cada nivel de una variable categórica.

La función `aggregate` recibe por parámetro un (sub)conjunto de datos, la variable categórica sobre la cual se va agrupar y la función que calcula la estadística. La razón por la cual usualmente la función recibe como parámetro un subconjunto y no toda la tabla de datos, es debido a que para algunas variables (como las categóricas) no tiene sentido el cálculo de una estadística definida solo para variables continuas. Con esto se resalta que se debe tener siempre presente el tipo de cada variable en el conjunto de datos.

El siguiente ejemplo muestra el uso de la función `aggregate` calculando el peso y la estatura promedio por género.

```
# Función aggregate

aggregate( personas[,c(1,2)], list(Generos=personas$genero), mean )
  Generos         estatura        peso
1         F       1.615000        51.00000
2         M       1.836667        94.66667
```

3.3.3. Correlaciones en R

Las correlaciones, son indicadores de qué tanto dos variables se relacionan entre sí. En análisis de datos, el cálculo de correlaciones es una forma rápida y sencilla de obtener un entendimiento inicial del conjunto de datos.

La función utilizada en R para el cálculo de correlaciones es la función `cor`. Esta función recibe por argumento las dos variables cuya correlación se desea calcular y el método de correlación. Si no se especifica explícitamente el método para calcular la correlación, la función asume el método de correlación de Pearson.

```
# Correlación de Pearson
cor(x,y)
```

A diferencia de otras funciones como `mean`, `sd` o `var`, la función `cor` tiene un tratamiento diferente de los valores faltantes. En lugar de utilizar el parámetro `na.rm=T` debe emplearse el parámetro `use="complete.obs"`, según se muestra a continuación.

```
cor(x, y, use="complete.obs")
```

Para utilizar métodos de cálculo del coeficiente de correlación, diferentes al método de Pearson, se utiliza el argumento `method` en la función `cor`. Este argumento puede tomar los valores spearman o kendall para utilizar los métodos de `Spearman` o `Kendall`, respectivamente.

```
# Correlación por método de Spearman
cor(x, y, method="spearman")
# Correlación por método de Kendall
cor(x, y, method="kendall")
```

También es posible calcular la covarianza de dos variables utilizando la función `cov`.

3.3.4. Funciones probabilísticas en R

Esta sección presenta las principales funciones provistas por R para realizar cálculos que involucren nociones de análisis combinatorio y distribuciones de probabilidad.

La función `choose(n, k)` es utilizada en R para calcular el coeficiente $\binom{n}{k}$. Debe recordarse que en el contexto del análisis combinatorio esta función se interpreta como el número de formas de seleccionar *k* números o ítems de un total de *n* disponibles.

```
# Cálculo de combinatoria
choose(45, 5)
[1]   1221759
```

Desde un punto de vista aplicado, una de las principales ventajas de R como un entorno estadístico, es tener la posibilidad de realizar cálculos combinatorios y trabajar, de manera relativamente rutinaria, con conceptos de teoría de probabilidad como variables aleatorias, distribuciones de probabilidad o funciones de densidad de probabilidad.

Un objetivo de este libro es exponer cómo se realizan cálculos que involucran estos conceptos en R. Si bien únicamente se realiza la exposición para una distribución continua (la distribución normal) y para una distribución discreta (la distribución binomial), los conceptos son extrapolables al resto de distribuciones disponibles en R.

En R, cuatro ítems fundamentales pueden ser calculados para cada distribución de probabilidad:

- Función de densidad de probabilidad (o probabilidad puntual, en el caso discreto). En el caso continuo, puede interpretarse como una medida de la probabilidad relativa de obtener un valor cercano a un valor *x*, mientras que en el caso discreto es la medida que indica la probabilidad de obtener exactamente el valor *x*, de ahí el nombre probabilidad puntual.

- Función de distribución acumulada. Se interpreta como la probabilidad de obtener un valor *x* o menor en una distribución dada.
- Función cuantil. Se define como la inversa de la función de distribución acumulada. El *p*-cuantil es el valor con la propiedad que existe una probabilidad *p* de obtener un valor igual o menor a él.
- Números pseudo-aleatorios. Generación de números pseudo-aleatorios a partir de una distribución de probabilidad.

Para cada distribución de probabilidad implementada en R, existe una función que calcula cada uno de los ítems anteriores, funciones que mantienen una notación consistente o convención en todas las distribuciones. Por ejemplo, la distribución binomial es nombrada en R como `binom`; las funciones para calcular los ítem anteriores son `dbinom`, `pbinom`, `qbinom` y `rbinom` (densidad, probabilidad, cuantil —debido al término en inglés quantile—, y pseudo-aleatorios —debido al término random—). De forma análoga, la distribución normal es denominada en R `norm`. Siguiendo la convención, las funciones para calcular los ítems son `dnorm`, `pnorm`, `qnorm` y `rnorm`, respectivamente.

A manera de ejemplo, se muestra en el siguiente segmento de código la forma de generar números pseudo-aleatorios basados en la distribución normal. La función a utilizar es `rnorm`, la cual recibe por parámetro la cantidad de números, la media y la desviación estándar. En caso de no suministrar a la función la media y la desviación estándar se asume una distribución normal estándar con media 0 y desviación estándar 1.

```
# Generación de números pseudo-aleatorios
# Distribución normal
rnorm(20)
[1]       -2.1231785      0.5694910       -0.6523139 ...  -0.7354254
[7]       -0.4793231              -0.2873925      -0.6943303  ...
-0.4367968
[13]      -0.7645333      -1.1893440      1.1032790 ...   -0.4948460
[19]      -0.7739264      -0.1588329


rnorm(10,mean=5,sd=1)
[1]       4.029394        5.776133        4.411990        5.9012088
...       7.045746
[9]       4.836043        3.953871
```

Funciones equivalentes para otras distribuciones de probabilidad teóricas comúnmente utilizadas se muestran en la tabla 3.12.

3.3.5. Estadística inferencial básica en R

La noción de muestra aleatoria es básica para la estadística inferencial y tiene que ver en gran parte con la generación de números aleatorios. Para lograr esto en el entorno R, una primera aproximación es utilizar la función `sample`, cuyos argumentos pueden variar, pero generalmente incluyen: un objeto de tipo vector de donde se obtendrá la muestra, el tamaño de la muestra, un indicador de si la muestra se realiza *sin reemplazo* (en cuyo caso el tamaño de la muestra no puede ser mayor al tamaño del vector) *o con reemplazo*, y las probabilidades de cada uno de los posibles eventos que no siempre son simétricas. Que una muestra se realice con reemplazo, implica que pueden existir valores duplicados en la muestra. Adicionalmente, cabe mencionar que por defecto la función `sample` realiza el proceso de muestreo sin reemplazo.

Tabla 3.12. Denominaciones de función de probabilidad en R

Función de Probabilidad	Denominación R
t	`t`
F	`f`
Binomial	`bincm`
Poisson	`pois`
Chi Cuadrado	`chisq`
Exponencial	`exp`
Weibull	`weibull`
Geométrica	`geom`
Hipergeométrica	`hyper`
Logística	`logis`
Beta	`beta`
Gamma	`gamma`

Fuente: Elaboración propia

Un ejemplo de la utilización de la función `sample` para generar una muestra de diez números del 1 al 1000 con reemplazo y con probabilidades simétricas se muestra a continuación:

```
# Generación de una muestra con repetición
sample(1:1000, 10, replace=T)
[1]   969   457   756   138   317   512   344   614   878   525
```

Una aplicación común del muestreo consiste en seleccionar un número aleatorio *n* de observaciones de un conjunto de datos. La forma de realizar esto en R, utilizando la función `sample` se muestra en el siguiente código, en el cual se hace uso de la función `nrow` para determinar el total de observaciones presentes en el conjunto:

```
# Muestreo de un data frame
muestras <- dataframe[sample(1:nrow(dataframe), n), ]
```

Al igual que en el ejemplo anterior, si se desea realizar el proceso de muestreo con repetición se invoca la función `sample` con el argumento `replace=T`.

Una vez se cuenta con un mecanismo para obtener muestras aleatorias, es posible realizar diferentes análisis estadísticos sobre estas, en particular, con pruebas de hipótesis (Sección 2.3). Siendo R un paquete de software estadístico no es de sorprender que existan funciones para calcular cada una de las pruebas estadísticas comunes. Una vez se determina cuál es la función en R que calcula una prueba, el ejercicio es establecer cuáles son los parámetros de la función adicionales a la variable `dataset$variable` y sobre todo cómo interpretar los resultados que R arroja. Las funciones R que realizan las diferentes pruebas estadísticas se resumen en la tabla 3.13.

Tabla 3.13. Pruebas de hipótesis comunes en R

Prueba	Función R	Parámetros
Shapiro-Wilk	`shapiro.test`	
Kolmogorox-Smirnov	`ks.test`	Parámetros de la distribución
t de una muestra	`t.test`	`mu,conf.level`
t de dos muestras	`t.test`	`var.equal`, Fórmula del modelo
t de muestras dependientes	`t.test`	`paired=T`
Wilcoxon de una muestra	`wilcox.test`	`mu`
Prueba *F*	`var.test`	Fórmula del modelo
Prueba de correlación	`cor.test`	

Fuente: Elaboración propia

3.3.6. Modelos lineales en R

Para implementar un modelo de regresión lineal en R se utiliza la función `lm`, que es la abreviación de *Lineal Model*, traducción literal en inglés de "modelo lineal". En su forma más genérica, la función requiere únicamente una *fórmula* para realizar la regresión y el conjunto de datos sobre los cuales se efectuará la misma.

```
# Regresión lineal
lm( formula, dataframe)
```

Como resultado de invocar la función `lm` se obtiene un objeto R cuyo tipo de dato es "lm". Este posee información adicional a los coeficientes de la regresión, en particular información como los residuales, el error estándar, el resultado de una prueba estadística *F* y otra información de interés. Toda esta información se puede obtener utilizando la función `summary` sobre el objeto de tipo "lm".

La fórmula involucrada en la regresión debe tener una sintaxis particular para que R la interprete como tal. Esta notación debe seguir la siguiente estructura: *dV ar ~ iV ar1 + iV ar2 + · +iV arn*. En este formato, *dV* ar representa la variable dependiente, mientras *iV ari* constituye una o muchas variables independientes, ya que la función `lm` acepta tantas variables independientes como sea necesario. El símbolo ~ debe interpretarse como "está descrita por", de forma que y ~ x debe leerse como "y está descrita por x".

3.3.7. Series de tiempo en R

Para la manipulación de series de tiempo, R cuenta con un tipo de objeto denominado `ts` (iniciales de *Time Series*). Un objeto R perteneciente a esta clase contiene las observaciones, el tiempo de inicio y la finalización de la serie e información sobre su *frecuencia o periodicidad* como, por ejemplo, mensual, trimestral, semestral o anual. Otras extensiones de la plataforma R permiten incluso manipular series de tiempo hora a hora o incluso minuto a minuto.

La función `ts` es la encargada de crear un nuevo objeto de tipo serie de tiempo. Recibe como argumento las observaciones de la variable, la frecuencia (1 para datos anuales, 4 para datos trimestrales, 6 para semestrales y 12 para mensuales) e información sobre el inicio y la finalización de la serie de tiempo.

```
# Creación de una serie de tiempo
ventas.st <- ts( obs,  start=c(2003,1), end=c(2004,12), frecuency=12 )
```

Existen funciones para consultar cada parámetro de la serie de tiempo: la función `start` permite obtener el inicio de la serie de tiempo; la función `end` permite obtener el final de la serie de tiempo, y la función `frequency`, la frecuencia o periodicidad de la serie de tiempo. Todas ellas reciben por parámetro un objeto de tipo `ts`.

```
# Funciones para manipulación de
# series de tiempo
start(ventas.st)
[1]   2003     1
end(ventas.st)
[1]    2004     12
frequency(ventas.st)
[1]    12
```

Para usar la función que permita obtener una serie de tiempo suavizada mediante la técnica de promedio móvil es necesario instalar el paquete `forecast` y cargarlo a la sesión de trabajo.

```
# Instalación del paquete forecast
install.packages ("forecast")
library (forecast)
```

Una de las varias funciones para obtener una serie de tiempo suavizada mediante la técnica de promedio móvil es la función `ma` —iniciales de *moving average* o promedio móvil—. Esta función recibe por parámetro la serie de tiempo original y el valor *k*. Debe recordarse que este valor es un número impar.

```
# Suavizado mediante promedio móvil
ventas.st.suavizada   <- ma(ventas.st,   7)
```

Para realizar la descomposición de una serie de tiempo en los componentes de estacionalidad, tendencia e irregularidades (error) se utiliza la función `stl`. Esta función recibe como argumentos una serie de tiempo y uno denominado `s.window` que suele tomar el valor "`period`", cuyo objetivo es indicar a la función que la componente estacional es la misma para todos los intervalos de muestreo.

```
# Descomposición de una serie de tiempo
ajuste <- stl( ventas.st, s.window="period" )
class(ajuste)
[1]    "stl"
```

Existen varias funciones en R para implementar modelos exponenciales. La instalación básica de R provee la función `HoltWinters`, pero en el paquete `forecast` existe una función más versátil denominada `ets`, que permite implementar otros modelos y que en general se considera más poderosa que la existente en la instalación básica.

La función `ets` tiene un formato en el que recibe dos argumentos: 1) la serie de tiempo y 2) el modelo, especificado con tres letras. La primera letra denota el tipo de error; la segunda el tipo de tendencia y la tercera letra el tipo de estacionalidad. Las letras permitidas son *A* para aditivo, *M* para multiplicativo, *N* para ninguno y *Z* para selección automática. Esta amplia versatilidad de la función permite que sea utilizada para cualquiera de los modelos exponenciales (*simple, holt, holt-winters*).

A manera de ejemplo, para un modelo exponencial simple, se utiliza la función `ets` con el parámetro `model="ANN"`. Esto quiere decir que el error (irregularidad) es aditivo y no hay componente de tendencia ni de estacionalidad, lo cual es consecuente con los supuestos del modelo.

```
# Modelo Holt-Winters
ajuste <- ets( st, model="ANN" )
```

La función `ets` realiza un ajuste del modelo y calcula los parámetros necesarios para efectuar el pronóstico como el valor de α en un modelo exponencial simple. Para hacer el pronóstico, se utiliza la función `forecast`, la cual recibe como parámetro el modelo ajustado y número *k* de valores futuros. Debido a que los modelos exponenciales suelen funcionar mejor para predicciones de corto plazo, este valor *k* no debería ser muy alto. De hecho, para valores diferentes a *k = 1* la función repite el valor calculado.

```
# Pronóstico de una serie de tiempo
forecast( ajuste, 1 )
```

Finalmente, es importante mencionar que si la función `ets` se invoca sin el parámetro `model` es porque `ets` está en capacidad de seleccionar automáticamente el modelo que mejor se ajuste a los datos.

```
ajuste <- ets( st )
```

3.4. Visualización en R

Parte fundamental de un proyecto de análisis de datos es la capacidad para comunicar los resultados de forma clara y efectiva. Debido a esto, los proyectos de análisis de datos cada vez más se apoyan en los aspectos de *visualización* de los datos, siendo consecuentes con que los seres humanos tienen la capacidad de extraer información útil a partir de gráficos, ya que estos son más intuitivos la mayoría de veces. No en vano la frase "una imagen vale más que mil palabras" es bastante popular, y de hecho en extremo precisa para describir su importancia en los proyectos de ciencia de datos, a tal punto que en ocasiones la estética de las visualizaciones y la información que provee puede llegar a tener una importancia equiparable a los análisis estadísticos (¡lo cual no quiere decir de ninguna forma que un excelente gráfico pueda camuflar un mal análisis!).

R provee facilidades para visualizar diferentes tipos de gráficos y tablas y funcionalidades para realizar gráficos personalizados con amplio control sobre la apariencia de los mismos. R permite crear gráficos elegantes e informativos de manera directa y con menor dificultad que en otros paquetes estadísticos. Sin embargo, esta amplia funcionalidad hace que no sea posible cubrir en este libro todas las funciones de R para realizar gráficas reduciendo la presentación a aquellas más utilizadas.

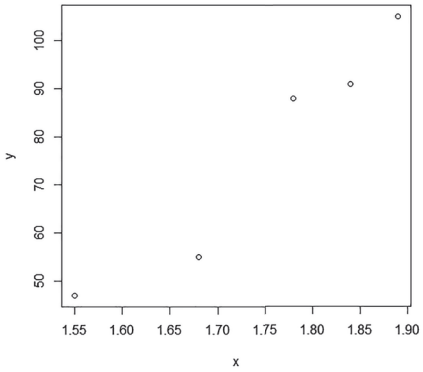
3.4.1. Gráficos x-y

El gráfico x-y estándar permite visualizar dos variables *x* e *y*, cada una de ellas con su respectiva etiqueta (que puede ser personalizada), así como adicionar títulos o subtítulos al gráfico. Este tipo de gráfico es muy utilizado para determinar la relación entre dos variables continuas, siendo *x* la variable explicativa e *y* la variable de respuesta.

La función `plot` es la función de R encargada de visualizar este tipo de gráfico, la cual recibe como parámetros —en su versión más sencilla— las dos variables (figura 3.3.) o parámetros adicionales, como el título del gráfico (`main`), el subtítulo (`sub`) o los nombres de las variables (`xlab/ylab`), entre otros tantos parámetros (figura 3.4.).

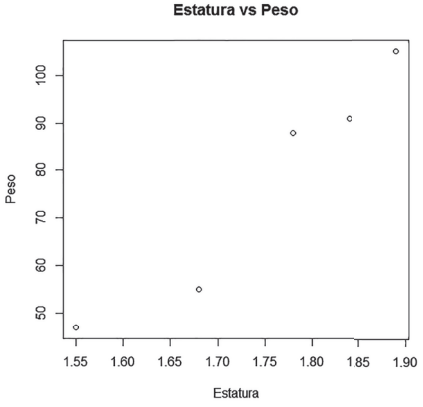
```
# Gráfico x-y simple
x <- personas$estatura
y <- personas$peso
plot(x,y)
# Gráfico x-y con título y etiquetas
plot(x,y,main="Estatura vs Peso", xlab="Estatura", ylab="Peso")
```

Figura 3.3. Gráfico xy simple con función plot



Fuente: Elaboración propia

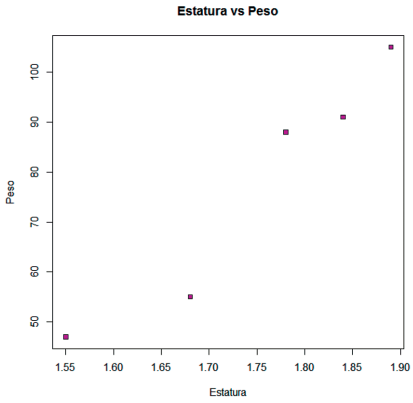
Figura 3.4. Gráfico xy simple con configuraciones adicionales



Fuente: Elaboración propia

También es posible personalizar el carácter utilizado en el gráfico (el cual por defecto es un círculo) haciendo uso de los parámetros `pch` y `bg`. El parámetro `pch` hace referencia al tipo de símbolo a utilizar, el cual puede ser un disco sin relleno, un triángulo, una cruz, una estrella u otros tantos. A la fecha de escritura de este documento, `pch` puede tomar valores entre 1 y 25. Si el caracter utilizado en el gráfico permite color de relleno (valores 21 a 25 en `pch`), `bg` representa el color de llenado del carácter, al cual se asigna una cadena de caracteres con el nombre del color. El nombre del color debe estar en idioma inglés, siendo valores válidos "green", "red", "blue", "cyan", "purple", "magenta" y otros más que se encuentran en la documentación de la función. Esta personalización puede verse en la figura 3.5.

Figura 3.5. Gráfico xy simple con configuraciones adicionales (2)



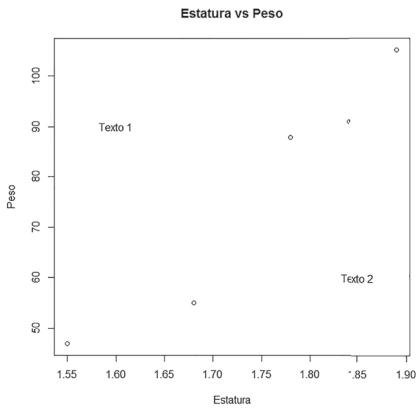
Fuente: Elaboración propia

En el modelo de gráficos utilizado por R, es posible dibujar puntos, líneas e incluso texto en la misma región de un gráfico que previamente se ha dibujado utilizando la función `plot`, como por ejemplo a la hora de comprobar visualmente si cualquiera de las gráficas se ajusta a alguna distribución de probabilidad particular.

Para dibujar texto en una región gráfica se utiliza la función `text`, que recibe por parámetro los puntos "x" e "y" y el texto que se desea imprimir. Es importante aclarar que los argumentos "x" e "y" están referenciados a la misma escala que previamente se utilizó en la función `plot`. La figura 3.6. muestra el resultado de ejecutar en R el siguiente código de ejemplo, en el cual debe notarse cómo los parámetros utilizados en la función `text` corresponden con las escalas utilizadas previamente en la función `plot`.

```
# Dibujo de texto
plot(x,y,main="Estatura vs Peso", xlab="Estatura", ylab="Peso")
text(1.60, 90, "Texto 1")
text(1.85, 60, "Texto 2")
```

Figura 3.6. Gráfico xy simple con texto

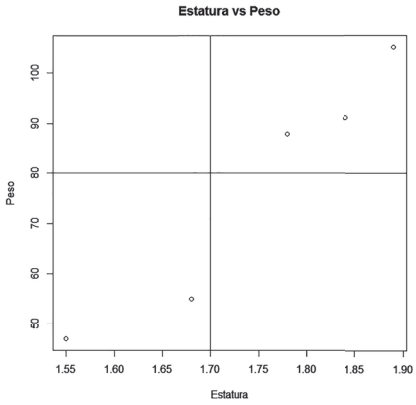


Fuente: Elaboración propia

Para dibujar una o varias líneas rectas sobre un gráfico puede utilizarse la función `abline`, la cual recibe como argumentos dos valores *a* y *b* y dibuja sobre la región de gráficos la línea $y = a + bx$. También es posible utilizar esta misma función para dibujar líneas verticales u horizontales, invocando la función con los parámetros `h` y `v`. Formas alternativas de dibujar líneas continuas se explicarán posteriormente.

```
# Dibujo de líneas rectas
abline(h=80,   v=1.70)
```

Figura 3.7. Gráfico xy simple con líneas sobrepuestas



Fuente: Elaboración propia

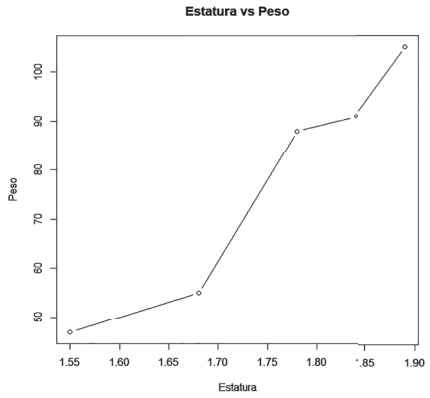
En ocasiones no se desean graficar únicamente los puntos (*x, y*) de la gráfica, sino también la conexión entre estos mediante segmentos de recta. Para lograr este objetivo, la función `plot` provee un argumento denominado type que logra el objetivo que se busca cuando el valor es "`b`", esto es, `type="b"`. La letra "`b`" es debido al término *both* en inglés, que significa "ambos" e indica que se van a graficar tanto líneas como puntos.

```
# Dibujo de líneas continuas
plot(x,y,main="Estatura vs Peso", xlab="Estatura",
  ylab="Peso", type="b")
```

Como es de esperarse, debido a las características de personalización que permiten los gráficos en la plataforma R, es posible cambiar el tipo de línea (sólida, punteada, intercalada, etc.) y su grosor. El parámetro `lty` especifica el tipo de línea, el cual es un número entero entre 1 y 6. Por su parte, el parámetro `lwd` expresa con un número entero un valor relativo al grosor predeterminado, de forma que un valor de 2 indica un grosor del doble del valor predeterminado, 3 el triple del grosor y así para otros valores. Mayor información de estas opciones de configuración se puede encontrar en la documentación de la función `plot`. La figura 3.9 ilustra cómo luce la figura 3.8 cuando se cambia el tipo y el grosor de la línea que conecta los puntos.

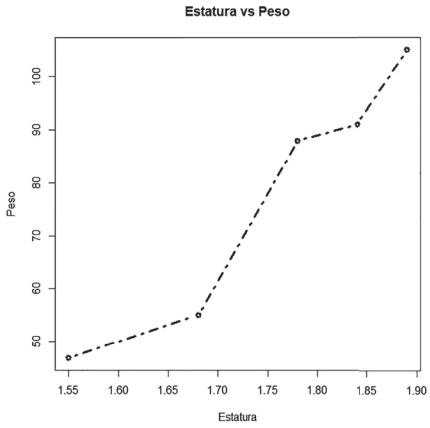
```
# Personalización de tipo y grosor de línea
plot(x,y,main="Estatura vs Peso", xlab="Estatura",
  ylab="Peso",   type="b",   lty=4,   lwd=3 )
```

Figura 3.8. Gráfico xy simple con líneas conectando los puntos



Fuente: Elaboración propia

Figura 3.9. Gráfico xy simple con líneas conectando los puntos. Configuración de tipo y grosor de línea



Fuente: Elaboración propia

Sin embargo, para que la apariencia de un gráfico con una línea conectando los distintos puntos sea la esperada, los datos del eje "x" deben estar *ordenados ascendentemente*. En este capítulo se introducirán otros tipos de gráficos comunes y útiles en análisis de datos.

3.4.2. Gráficos de barras

Son herramientas útiles para visualizar *valores agregados* (totales por categoría), por ejemplo, de una variable. Consiste en un conjunto de rectángulos o "barras" donde la altura de cada barra corresponde con el valor de la variable.

En la plataforma R, la función utilizada para realizar un gráfico de barras es la función `barplot`, que recibe por parámetro la salida de una función de agregación, esto es, una función que genera un único valor a partir de una colección tal como la función `sum` o la función `count` (la cual se encuentra en el paquete `dplyr`).

Existen múltiples formas de tratar con las funciones de agregación, pero quizás la más conveniente al trabajar con los gráficos de barras es la función `by`, la cual es análoga a la instrucción `GROUP BY` utilizada en SQL. Es usada para realizar operaciones de agregación sobre una variable que puede tomar diferentes valores. Toma por parámetro la tabla de datos, la variable en la cual se quiere aplicar la función de agregación (variable a agrupar), y la función de agregación a aplicar a cada subconjunto. La función también puede agregar múltiples variables pasando una lista en lugar de una única variable como segundo argumento de la misma.

A manera de ejemplo de cómo utilizar la función `by`, considérese el problema de totalizar cuántas observaciones de cada género se tienen en un data frame de personas, según la estructura que se ha venido manejando en el capítulo. En este caso, la función se invoca con los parámetros *by (personas, genero, count)*, siendo "personas" la tabla de datos, "genero" la variable cuyos subconjuntos se desean agregar y "count" la función de agregación.

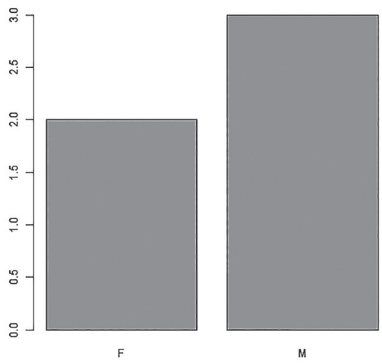
```
# Ejemplo de la función by para totalización
by(personas,genero,count)
[1] 2
------------------------------------------------------------
[1] 3
```

En este caso particular, también puede utilizarse la función `table` para el mismo fin, ya que es una función de agregación. Sin embargo, en general puede llegar a ser más conveniente utilizar la función `by`.

```
# Ejemplo de la función table para agregación
table(personas$genero)
F        M
2        3
```

La función `barplot` utiliza la salida generada por una función de agregación para producir la gráfica de barras. En el ejemplo anterior, la altura de cada barra corresponde a cuántas observaciones tiene cada género, según se muestra en la figura 3.10.

Figura 3.10. Gráfico de barras utilizando la función "barplot"



Fuente: Elaboración propia

3.4.3. Histogramas

Son gráficos que ilustran la frecuencia con la que aparece cada valor de una variable, se utilizan cuando se desea obtener una idea de la forma de una distribución de probabilidades, dividen el eje x de la gráfica en compartimentos o divisiones e ilustran un conteo de cuántas observaciones se encuentran en esa división, usualmente como un gráfico de barras.

En el entorno R, el comando utilizado para dibujar un histograma es el comando `hist`, que recibe por parámetro la variable cuyo histograma se desea graficar (es la versión más sencilla del comando), así como otros parámetros que permiten la personalización de la gráfica.

```
# Histograma
x <- rnorm( 100 )
hist(x)
```

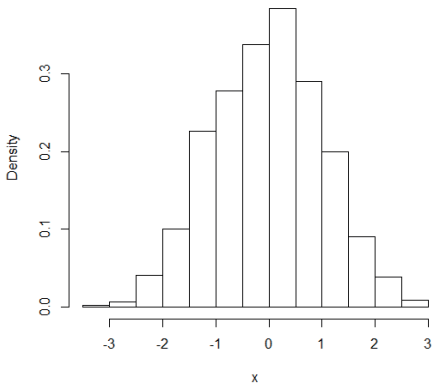
El argumento `breaks` permite determinar el número de divisiones o compartimentos del histograma, y puede ser especificado como un número o como un vector, lo cual es necesario cuando cada división tiene una escala diferente. A manera de ejemplo (tomado de [3]) considérese un conjunto de datos con valores que representan el número de accidentes en función de diversos grupos de edad. Estos conteos son realizados sobre los grupos de edad 0-4, 5-9, 10-15, 16, 17, 18-19, 20-24, 25-59 y 60-79 años. Como puede observarse no hay uniformidad en la escala de los grupos de

edad, ya que en algunos casos solo se contempla un año, mientras en otros existen hasta 19 de diferencia.

Edades	Accidentes
0-4	28
5-9	46
10-15	58
16	20
17	31
18-19	64
20-24	149
25-59	316
60-79	103

```
# Histograma con escala no uniforme
accidentes <- c(28, 46, 58, 20, 31,
  64, 149, 316, 103)
med.edades <- c(2.5, 7.5, 13, 16.5, 17.5,
  19, 22.5, 44.5, 70.5)
limites <- c(0, 5, 10, 16, 17, 18, 20,
  25, 60, 80)
```

Figura 3.11. Ejemplo de histograma

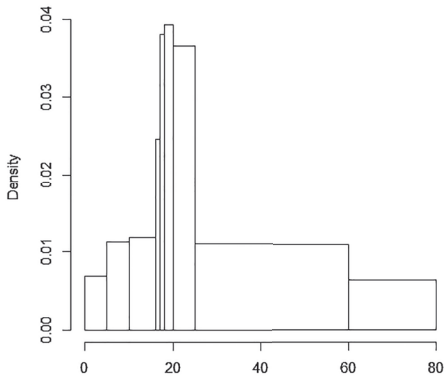


Fuente: Elaboración propia

El enfoque en este caso es generar un conjunto relevante de "observaciones" para cada grupo de edad, representándolo con su punto medio. Después de esto, y con el objetivo de graficar el histograma, se genera un nuevo vector con los límites de cada división. Finalmente, se grafica el histograma utilizando la función `hist` con el argumento `breaks`.

```
hist( rep( med.edades, accidentes), breaks=limites)
```

Figura 3.12. Ejemplo de histograma con divisiones no uniformes



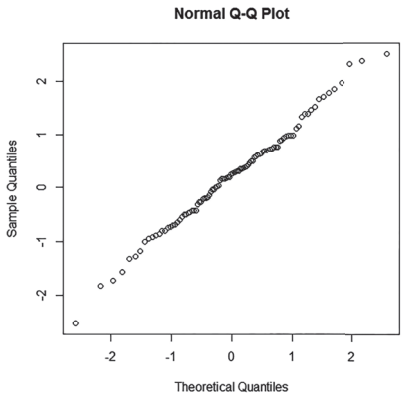
Fuente: Elaboración propia

3.4.4. Gráficos *Q-Q*

Los gráficos "cuantil vs. cuartil" (o gráficos Q-Q) tienen como propósito determinar si los datos que se grafican provienen de una distribución determinada, en cuyo caso se obtiene una línea recta. Son utilizados especialmente para visualizar de forma gráfica si una variable está distribuida normalmente, es decir, si sus datos provienen de una distribución normal (con cualquier varianza y media). Para este propósito particular, se utiliza la función `qqnorm`, que recibe como parámetro la variable. Un ejemplo de gráfico Q-Q obtenido con esta función se muestra en la figura 3.13.

```
# Gráfico Q-Q
x <- rnorm(100)
qqnorm(x)
```

Figura 3.13. Ejemplo de gráfico Q-Q

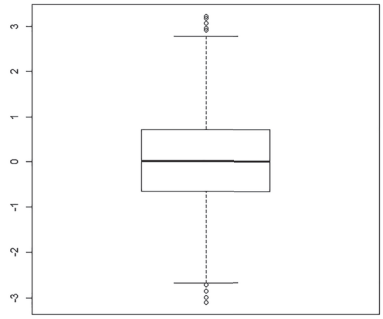


Fuente: Elaboración propia

3.4.5. Gráficos de caja

También conocido como "diagrama de cajas y bigotes", es un resumen gráfico de una variable o de una distribución y representa de manera visual la gran mayoría de estadísticas descriptivas que se calculan en la función `summary`. Un ejemplo de este gráfico se muestra en la figura 3.14.

Figura 3.14. Ejemplo de gráfico de caja



Fuente: Elaboración propia

La interpretación de un gráfico de caja es como sigue: la caja en la mitad del gráfico tiene varios "ejes", que aproximadamente representan los cuantiles: el eje inferior de la caja representa el primer cuartil del 25 % *Q*1; el eje de la mitad representa la mediana (cuartil del 50 % o *Q*2), y el eje superior representa el tercer cuartil *Q*3. Las líneas por fuera de la caja (los "bigotes") muestran las observaciones con los valores máximo y mínimo que residen dentro de una distancia de *1.5 veces* el rango intercuartil.

Las observaciones que quedan por fuera de estas líneas son "valores extremos" y se muestran separadamente.

A manera de ejemplo, supóngase que una variable tiene un rango intercuartil de *IQR* =3,909 y que sus cuartiles son *Q*1 = 10,995 y *Q*3 = 14,904. Con estos datos se considerarán valores extremos a todos aquellos valores menores que 10,995 − 1,5 * 3,090 = 5131,5 o mayores que 14,904 + 1,5 * 3,909 = 20767,5.

3.4.6. Múltiples gráficos

Es posible lograr un nivel de personalización de los gráficos en R bastante avanzado, teniendo la posibilidad de controlar muchos detalles de un gráfico, como el tipo y grosor de línea, tipo y tamaño de fuente, colores, tamaño del gráfico, regiones del gráfico, división en subfiguras, etc. Si bien algunas de estas opciones pueden ser controladas mediante los argumentos de las funciones `plot`, `hist`, `box` y similares (dependiendo del tipo de gráfico), otras requieren manejo diferente, este es el caso si se requiriera la generación de múltiples gráficos.

Para colocar muchos elementos en la misma gráfica, es necesario utilizar la función `par`, la cual es una función versátil de R utilizada para la configuración de márgenes, anchos, líneas, colores y tamaños a un nivel "global". Como es de esperarse, todos estos aspectos de visualización se configuran a través de parámetros, y en el caso particular de los múltiples gráficos, estos son `mfrow` y `mfcol`.

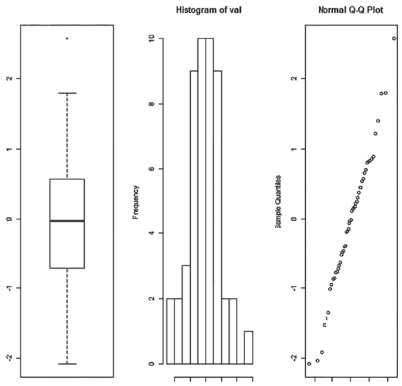
Cuando se invoca la función `par` con el parámetro `mfrow`, se especifica que los gráficos deben quedar uno al lado del otro, en un determinado número de columnas *n*. Por ejemplo, si se desea una única fila y tres columnas, el valor que debe tomar el parámetro `mfrow` es `mfrow=c(1,3)`. De forma análoga, el parámetro `mfcol` es utilizado cuando se desean gráficos que se muestren uno encima del otro. En este caso el parámetro toma el valor `mfcol=c(m,1)`, donde *m* representa el número de gráficos que se desean apilar.

A manera de ejemplo, se elaborará un gráfico que muestre simultáneamente una gráfica de caja, un histograma y una gráfica Q-Q de una variable, organizados uno al lado del otro. El código R para lograr este objetivo es el siguiente:

```
# Múltiples gráficos
val <- rnorm(50)
par(mfrow=c(1,3))
boxplot(val)
hist(val)
qqnorm(val)
par(mfrow=c(1,1))
```

El gráfico resultante del anterior ejemplo se muestra en la figura 3.15. También es importante aclarar el porqué de la última línea en el código del ejemplo anterior; esta línea es necesaria para que el entorno gráfico retorne a su funcionamiento por defecto de un gráfico a la vez, característica que fue alterada al utilizar la función `par`.

Figura 3.15. Ejemplo de varios gráficos



Fuente: Elaboración propia

Referencias bibliográficas

[1] G. J. Myatt y W. P. Johnson, *Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining*. Second Edition. New Jersey: John Wiley and Sons Inc., 2014. [En línea]. Disponible en: https://amzn.to/2Gclzte

[2] M. J. Crawley, *Statistics: An introduction using R*. Second Edition, John Wiley and Sons Inc., 2015. [En línea]. Disponible en: http://bit.ly/2JLUuhi

[3] R. I. Kabacoff, *R in Action*. Second Edition, Manning Publications Co., 2015. [En línea]. Disponible en: http://bit.ly/2LVVkuO

[4] P. Dalgaard, *Introductory Statistics with R*. Second Edition, New York: Springer, 2008. [En línea]. Disponible en: http://bit.ly/30yYw3k

[5] B. S. Everitt and T. Hothorn, *A Handbook of Statistical Analyses Using R*. Second Edition, CRC Press, 2010. [En línea]. Disponible en: http://bit.ly/2XZoPlR

[6] D. Nolan and D. Temple Lang, *Data Science in R: a Case Studies Approach to Computational Reasoning and Problem Solving*. 1st Edition, CRC Press, 2015. [En línea]. Disponible en: http://bit.ly/30wsYew

[7] A. B. Downey, *Think Stats. Exploratory Data Analysis*. O'Reilly Media Inc. 2015. [En línea]. Disponible en: https://amzn.to/2LVjh5g

[8] S. Stowell, *Using R for Statistics*. Apress, 2014. [En línea]. Disponible en: http://bit.ly/2YUXeiG[9]

[9] J. M. Quick, *Statistical Analysis with R. Beginner's Guide. Packt Publishing, 2010.* [En línea]. Disponible en https://amzn.to/2XM3soM

4. Fuentes y preparación de datos en R

Para aprender el máximo posible de cualquier situación o experiencia, se necesita recoger información desde el mayor número de puntos de vista posibles.

—John Grinder

Este capítulo trata del proceso referente a la obtención de datos, se asume que los datos necesarios para resolver el problema de análisis existen en algún lugar, en alguna forma, sean estos descargables de una página web, de una base de datos que puede ser consultada o ser accesibles a través de una Interfaz de Programación de Aplicaciones o *Application Programming Interface* (API), además de archivos en varios formatos, comprimidos en algunos casos.

En ciertas ocasiones los datos están disponibles de forma pública, pero no se encuentran directamente descargables o no cuentan con una API para obtenerlos; aún en este caso, es posible obtenerlos mediante un proceso conocido como *scrapping*, utilizando "robots" que navegan sitios web y sus hipervínculos. En muchos casos, los datos tienen formato de texto, por lo que no es de extrañar que la mayoría de herramientas utilizadas en el análisis estén preparadas para procesar este tipo de datos.

4.1. Archivos

4.1.1. Archivos de texto plano

Uno de los formatos más utilizados para representar un conjunto de datos es el de Valores Separados por Coma o *Comma-Separated Values* (CSV). En este formato, es posible representar directamente un data frame donde cada fila del archivo corresponde a una observación. Las variables de esta son espaciadas mediante un *símbolo separador*, el cual usualmente es el símbolo de la coma (","), aunque también es posible encontrar como separador el punto y coma (";"), una tabulación (espacios consecutivos) u otros caracteres como "`/`" o "`#`". En este formato es común que la primera fila tenga los nombres de las variables, también conocido como *encabezado*, con lo cual el archivo CSV genérico correspondiente al data frame 2.1 luciría como sigue:

```
  x1, x2, x3, ··· , xp
x11, x12, x13, ··· , x1p
x21, x22, x23, ··· , x2p
x31, x32, x33, ··· , x3p
.
.
.
xn1, xn2, xn3, ··· , xnp
```

Para cargar un archivo externo en formato CSV en un entorno de trabajo R se utiliza la función `read.csv`, la cual recibe por parámetro un nombre de archivo y devuelve por resultado un objeto de tipo data frame. La función asume que el archivo cuyo nombre se pasa como argumento de la función se encuentra en el directorio de trabajo actual del entorno R; en caso de no estar en este directorio, el argumento de la función `read.csv` debe referenciar la ruta completa del archivo.

```
# Lectura de un archivo en formato CSV
datos <- read.csv("ejemplo.csv")
```

Existen ligeras variaciones o situaciones que pueden llegar a ocurrir al leer un archivo de texto plano en formato CSV; la primera de ellas tiene que ver con el hecho de que el archivo no tenga los distintos encabezados o nombres de las variables, las cuales R asume como parte del formato. En este caso, la función `read.csv` requiere un nuevo argumento denominado "header" que toma un valor falso para indicar que los encabezados no se encuentran presentes. En estas circunstancias, las variables adquieren nombres genéricos *V1, V2*, etc, comportamiento que puede ser cambiado con el argumento `col.names` que toma como valor un vector con los nuevos nombres de las variables.

A manera de ilustración, supóngase que el archivo `ejemplo.csv` no tiene los encabezados y se quieren renombrar sus variables a "Nombre", "Apellido" y "Genero". El código R para lograr esto se muestra a continuación.

```
# Lectura de un archivo CSV sin encabezados
datos <- read.csv("ejemplo.csv", header=FALSE,
      col.names=c("Nombre", "Apellido", "Genero"))
```

Una segunda situación tiene que ver con los datos faltantes, ya que si bien algunos archivos pueden representar este hecho con un campo vacío, otros pueden utilizar algún símbolo especial para lo mismo. En este caso, el argumento `na.strings` debe ser pasado a la función `read.csv` indicando los caracteres utilizados para representar datos faltantes.

```
# Lectura de un archivo CSV con datos faltantes
datos <- read.csv("ejemplo.csv", na.strings=".")
```

Para aquellos archivos que no utilizan como separador de valores el símbolo "," (coma), R provee la función `read.table`, que se usa de forma similar a `read.csv`; el argumento `sep` es aplicado para indicar el símbolo manejado como separador, según se muestra en el siguiente ejemplo, donde el símbolo "`/`" es utilizado como separador.

```
# Lectura de un archivo CSV con separador no estándar
datos <- read.table("ejemplo.csv", sep="/", header=TRUE)
```

Una vez se tiene el archivo en un objeto de tipo data frame es posible manipularlo, tal como se explica en la sección 3.2.6, bien sea utilizando la notación de "brackets" o la notación `dataframe$variable`.

De forma análoga, R provee una función `write.csv` para exportar una tabla de datos a un archivo en formato CSV. La función recibe por parámetro la tabla de datos a exportar y el nombre del archivo al cual exportar; como resultado de invocar la función no se produce ninguna salida en la consola de R. Cabe mencionar que si existe un archivo con el mismo nombre, este será sobrescrito sin ninguna advertencia por parte de R, por lo que es importante validar que no se sobrescriba algún archivo vital del cual no se tenga respaldo.

```
# Escritura de un archivo CSV
write.csv(datos, "ejemplo_escritura.csv")
```

Sin embargo, al exportar de esta forma se crea una columna adicional que contiene el número de la observación, situación que no siempre es deseable. Para prevenir esto, se debe invocar la función con un argumento adicional `row.names`, al cual se le debe asignar un valor `F` o falso.

```
# Escritura de un archivo CSV sin número de
# observación
write.csv(datos, "ejemplo_escritura.csv", row.names=F)
```

4.1.2. Archivos JSON

JavaScript Object Notation (JSON) es un estándar de almacenamiento e intercambio de datos frecuentemente encontrado por las aplicaciones web para proveer datos a los desarrolladores de aplicaciones. Usualmente es el lenguaje de intercambio en muchas aplicaciones orientadas a servicios, esto es, que definen *interfaces* para que los

programadores o los programas se comuniquen con estas aplicaciones, bien sea para consultar datos o para actualizarlos. Si bien el formato tiene sus orígenes en el lenguaje JavaScript, este es independiente del mismo y puede ser interpretado por cualquier lenguaje de programación, incluyendo R. A manera de ejemplo de la forma como luce un formato en archivo JSON, considérese el siguiente conjunto de datos:

```
          estatura       peso    genero
1 1.      68             55      F
2 1.      84             91      M
3 1.      78             88      M
4 1.      55             47      F
5 1.      89             105     M
```

El archivo JSON cuya información es equivalente al anterior conjunto de datos es el siguiente:

```
[
  {
  "estatura": 1.68,
  "peso": 55,
  "genero": "F"
 },
 {
  "estatura": 1.84,
  "peso": 91,
  "genero": "M"
 },
 {
  "estatura": 1.78,
  "peso": 88,
  "genero": "M"
 },
 {
  "estatura": 1.55,
  "peso": 47,
  "genero": "F"
 },
 {
  "estatura": 1.89,
  "peso": 105,
  "genero": "M"
 }
 ]
```

Si bien la estructura general del lenguaje es bastante intuitiva, algunos aspectos particulares de su sintaxis merecen comentarios aclaratorios. El primero a resaltar del formato JSON es la importancia de los paréntesis y las llaves, en particular los símbolos "[", "]" "" y "". Estos funcionan como sigue:

- Las llaves "" y "" son utilizadas para delimitar cada objeto u observación. A su vez, cada objeto puede contener otros objetos, arreglos, pares llave/valor u otras colecciones.
- Los paréntesis cuadrados "[" y "]" delimitan *arreglos*, es decir, secuencias ordenadas de objetos o valores.

Los datos en JSON corresponden a pares *llave-valor*, esto es, a objetos que pueden ser accedidos por un nombre o identificador denominado *llave* y a su valor como tal, el cual puede ser un número, una cadena de caracteres, un valor lógico, un arreglo, otro objeto o incluso un valor desconocido o `null`. Los valores se separan de las llaves por el símbolo :, mientras los objetos se separan uno del otro mediante el símbolo ,. Las llaves y todos aquellos valores de tipo cadena de caracteres van encerrados en los caracteres "" y "".

Existen al menos tres paquetes en R para importar, exportar y manipular archivos JSON: `rjson, RJSONIO` y `jsonlite`. Si bien la conversión entre objetos JSON y objetos R en algunos casos no es tan directa, la librería `jsonlite` es quizás la que más ha trabajado en estos aspectos de las tres y es por ello que será la librería utilizada en este documento.

```
# Importar librería jsonlite
install.packages("jsonlite")
library(jsonlite)
```

Las dos funciones principales a utilizar son aquellas para convertir un objeto R de tipo data frame al formato JSON y aquella que realiza el proceso inverso. La primera función es `toJSON`, que recibe como argumento el objeto de tipo data frame y posiblemente el argumento `pretty = TRUE` para que los datos queden con los niveles de identación correctos. La segunda función es `fromJSON`, que recibe un archivo en formato JSON y retorna un objeto de tipo data frame equivalente. Como ejemplo, supóngase la existencia de un objeto data frame denominado `personas` y de un archivo `personas.json`. Las anteriores funciones se utilizan como sigue:

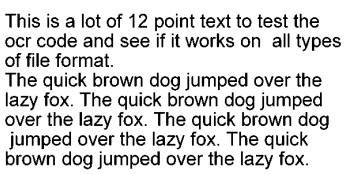
```
# Conversión de un data frame a JSON
personas.json <- toJSON(personas, pretty=TRUE)
# Conversión de JSON a data frame
personas.json <- fromJSON("personas.json")
```

Una vez se tiene el contenido de un archivo JSON en una estructura de R como data frame pueden realizarse las diferentes manipulaciones a las que haya lugar.

4.1.3. Imágenes

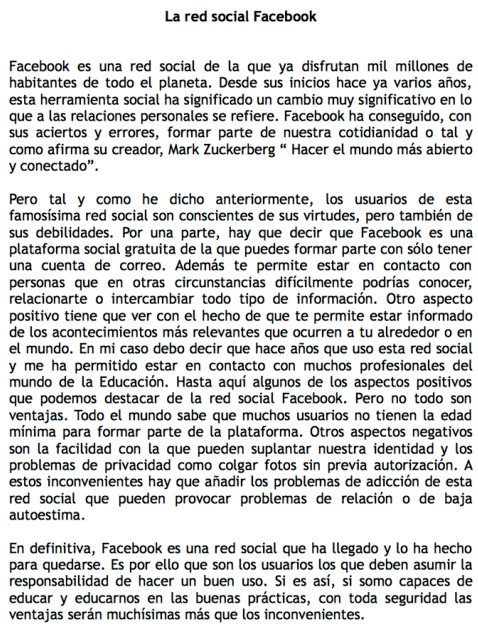
De los múltiples tipos de archivos que se encuentran en la red global internet, entre los más frecuentemente encontrados están aquellos de tipo *imagen*. Las imágenes pueden estar en diferentes formatos, siendo `.jpg`, `.png` o `.tiff` los más populares. En este libro son de particular interés aquellas imágenes que contienen texto, como las imágenes mostradas en las figuras 4.1 y 4.2.

Figura 4.1. Imagen con texto en idioma inglés



Fuente: Elaboración propia

Figura 4.2. Imagen con texto en idioma español



Fuente: Elaboración propia

El proceso de extraer el texto de documentos, como las imágenes mostradas o documentos digitalizados, se conoce como *Reconocimiento Óptico de Caracteres* u *Optical Character Recognition* (OCR). Reconocer patrones visuales, si bien es un proceso natural e intuitivo para los seres humanos, es increíblemente complicado para las máquinas, especialmente si la imagen contiene ruido u otros artefactos; de hecho, la exactitud de los resultados depende de la calidad de la imagen. El porqué este proceso es complejo involucra muchos factores, pero quizás uno de los más relevantes es que los seres humanos utilizan todo su conocimiento previo del lenguaje para inferir las ideas y entenderlas sin tener que realizar el proceso letra por letra o palabra por palabra, proceso que sí deben realizar las máquinas (a la fecha). También es asombrosa la capacidad de los seres humanos para reconocer los símbolos del lenguaje en diferentes formas, tipos, tamaños y colores, características que dificultan el proceso a las máquinas.

La librería en R que se va a utilizar en este libro para el proceso de reconocimiento óptico de caracteres se llama `tesseract`, software de código abierto para este proceso. La forma de instalar la librería se muestra a continuación:

```
# Instalación de la librería tesseract
install.packages("tesseract")
library(tesseract)
```

La función del paquete `tesseract` utilizada para realizar el proceso de reconocimiento óptico de caracteres se denomina `ocr` y recibe por parámetro en su versión más simple una imagen en alguno de los formatos de imagen populares. La función retorna un objeto de tipo cadena de caracteres.

```
# Reconocimiento óptico de una imagen
texto <- ocr("texto-ejemplo-ingles.jpg")
class(texto)
[1]   "character"
```

El conocimiento previo del lenguaje, necesario para lograr un proceso de reconocimiento óptimo con adecuados niveles de precisión se encuentra en unos archivos con *datos entrenados*, utilizados por `tesseract` para dar contexto a las palabras. Esta información se usa, por ejemplo, para diferenciar el número 1 de la letra "l" que, en ocasiones, lucen exactamente iguales al ojo humano. Este archivo para idioma inglés, idioma por defecto, se denomina `eng.traineddata`. El porqué este es el idioma por defecto y el que ofrece mejores resultados es consecuencia directa de que la gran mayoría de investigación y ciencia fundamental para los procesos de OCR y de procesamiento de lenguaje natural se han desarrollado en idioma inglés.

A manera de ejemplo, se puede observar que el texto de la figura 4.1 es decodificado casi perfectamente.

```
# Reconocimiento óptico de una imagen
texto <- ocr("texto-ejemplo-ingles.jpg")
cat(texto)

This is a lot of 12 point text to test the
cor code and see if it works on all types
of file format.
The quick brown dog jumped over the
lazy fox. The quick brown dog jumped
over the lazy fox. The quick brown dog
jumped over the lazy fox. The quick
brown dog jumped over the lazy fox.
```

Considérese a manera de otro ejemplo, el texto mostrado en la figura 4.3. En este caso el texto es decodificado perfectamente a partir de la imagen. Se enfatiza una vez más en que la librería está altamente entrenada y probada para el inglés, idioma para el cual funciona con excelentes niveles de exactitud.

Figura 4.3. Otra imagen de ejemplo con texto en idioma inglés

When in the Course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the Laws of Nature and of Nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

```
# Reconocimiento óptico de una imagen
texto <- ocr("texto-ejemplo-ingles-2.jpg")
cat(texto)

When in the Course of human events, it
becomes necessary for one people to dissolve
the political bands which have connected them
with another, and to assume among the
powers of the earth, the separate and equal
station to which the Laws of Nature and of
Nature's God entitle them, a decent respect to
the opinions of mankind requires that they
should declare the causes which impel them to
the separation.
```

Dado que no es el idioma por defecto, es necesario descargar los datos entrenados para el lenguaje español. Esto se realiza con la función `tesseract_download`, que recibe por parámetro el lenguaje cuyos datos entrenados se desean descargar. Para el caso del idioma español, el parámetro es `"spa"`.

```
# Descarga de datos de entrenamiento para
# idioma español
tesseract_download("spa")
```

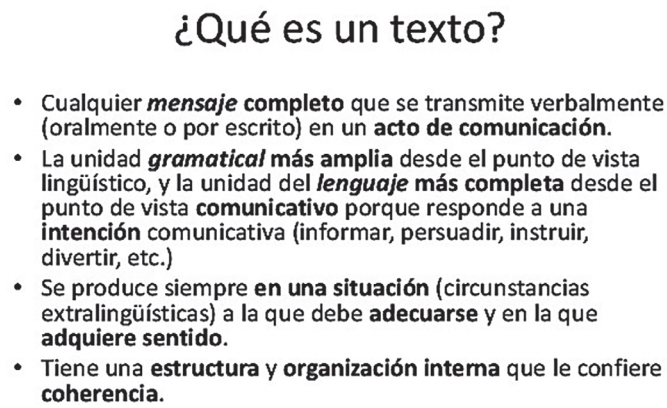
Una vez se descargan los datos de entrenamiento para el idioma español, es posible realizar el proceso de decodificar las imágenes con la función `ocr`. Sin embargo, es necesario indicarle explícitamente a la función que el idioma que se va a utilizar es el español, hecho que se indica con el parámetro `engine = tesseract(language = "spa")`.

```
# Reconocimiento óptico de una imagen
# en lenguaje español
texto <- ocr("texto-ejemplo-3.jpg",
          engine = tesseract(language = "spa") )
cat(texto)

La red social Facebook
Facebook es una red social de la que ya disfmtan ...
habitantes de todo el planeta. Desde sus inicios ...
esta herramienta social ha significado un cambio ...
que a las relaciones personales se refiere. Facebook
sus aciertos y errores, formar parte de nuestra ...
como afirma su creador, Mark Zuckerberg " Hacer ...
y conectado".
...
En definitiva, Facebook es una red social que ha ...
para quedarse. Es por ello que son los usuarios ...
responsabilidad de hacer un buen uso. Si es así, si
educar y educarnos en las buenas prácticas, con ...
ventajas serán muchísimas más que los inconvenientes.
```

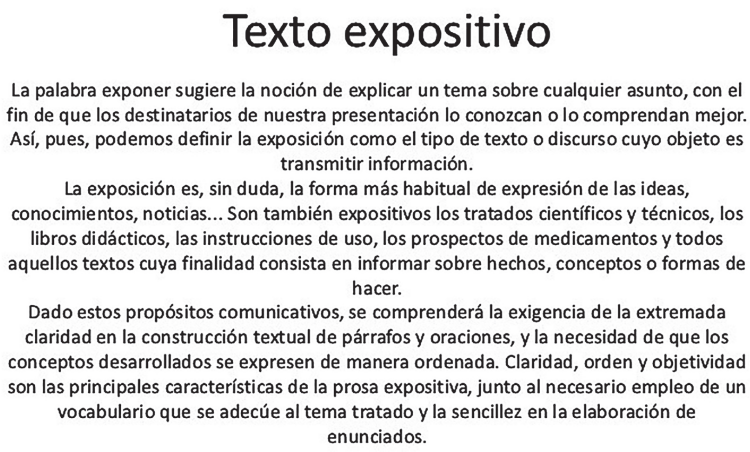
Para el caso de la figura 4.2, mostrado en el ejemplo anterior, se observa que los resultados son aceptables. Sin embargo, vale la pena explorar las limitaciones de la librería en el caso del idioma español, a medida que el texto de la imagen adquiere otros formatos. Ejemplos de este estilo de texto se encuentran en las figuras 4.4 y 4.5.

Figura 4.4. Ejemplo de imagen con texto en idioma español



Fuente: Elaboración propia

Figura 4.5. Ejemplo de imagen con texto en idioma español



Fuente: Elaboración propia

Los resultados obtenidos al ejecutar la función `ocr` sobre el texto de la figura 4.4 son los siguientes:

```
# Reconocimiento óptico de una imagen
# en lenguaje español

texto <- ocr("texto-ejemplo-1.jpg",
          engine = tesseract(language = "spa") )
cat(texto)
```

```
¿Qué es un texto?
— Cualquiermensnje completo que se transmite verbalmente
(oralmente o por escrito) en un acto de comunicación.
— La unidad gramatical más amplia desde el punto de vista
Hng...stíco, y la unidad del lenguaje más completa desde e)
punto de vista comunicativo porque responde a una
intención comunicativa (informar, persuadir, ínstruír,
dívertír, etc.)
' Se produce siempre en una situación (circunstancias
extralíng...5tícas) a ¡a que debe adecuarse y en ¡a que
adquiere sentido.
— Tiene una estructura y organización interna que le confiere
coherencia.
```

Que en algunos casos la exactitud de los textos decodificados mediante el proceso de reconocimiento óptico de caracteres no sea el ideal, no debería tomarse como aspecto negativo o indicador de que la librería `tesseract` no cumple su función para el idioma español. Por el contrario, sea esta breve exploración suficiente para motivar al lector a investigar y mejorar las técnicas de procesamiento de lenguaje natural en español, campo con amplia teoría y práctica por desarrollar y que, al día de hoy, se encuentra bastante lejano en el estado del arte de su contraparte de habla inglesa.

4.1.4. Archivos PDF

Los archivos Formato de Documento Portable o *Portable Document Format* (PDF) son un formato estándar (ISO 32000-1) de almacenamiento orientado a documentos digitales. Son de tipo compuesto, lo que quiere decir que constan de imágenes en varios formatos: texto, elementos multimedia, enlaces, miniaturas y otros componentes.

Es un formato extendido para intercambio de documentos, utilizado por empresas y gobiernos y en general cualquier organización o individuo, a tal punto que se considera un *estándar de facto* cuando se piensa que un documento está listo para imprimir exactamente como se muestra en el archivo. Si bien inicialmente los archivos se consideraban de solo lectura y se requería una aplicación especial para editarlos, hoy es común encontrar aplicaciones, muchas de código abierto, para editar directamente los documentos PDF.

El paquete en R para realizar las diversas funcionalidades de procesamiento de un documento PDF se denomina `pdftools`.

```
# Instalación de pdftools
install.packages("pdftools")
library(pdftools)
```

Una vez se instala el paquete `pdftools` es posible acceder al contenido de un archivo PDF. La función para obtener el contenido de un archivo PDF en la librería `pdftools` es `pdf_text`, que recibe como argumento el archivo a interpretar y retorna un objeto de tipo cadena de caracteres.

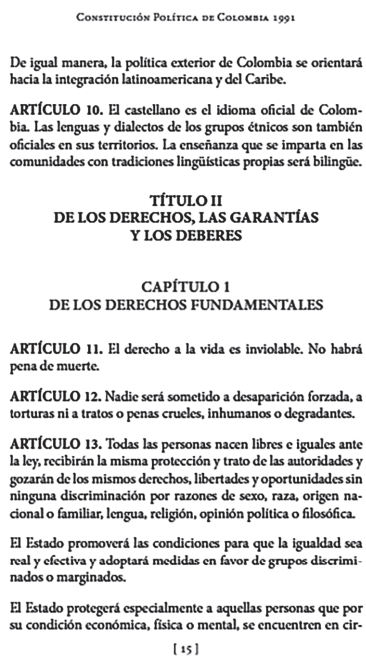
```
# Lectura de una archivo PDF
contenido <- pdf_text("Constitucion-1991.pdf")
class(contenido)
[1] "character"
```

A manera de ejemplo, se desea leer el contenido de un documento PDF con el texto completo de la Constitución Política de la República de Colombia de 1991. Parte del resultado de este proceso utilizando la función `pdf_text` es el siguiente:

```
contenido <- pdf_text("Constitucion-1991.pdf")
cat(contenido[15])
          Constitución Política de Colombia 1991
De igual manera, la política exterior de Colombia se orientará
hacia la integración latinoamericana y del Caribe.
ARTíCULO 10. El castellano es el idioma oficial de Colom-
bia. Las lenguas y dialectos de los grupos étnicos son también
oficiales en sus territorios. La enseñanza que se imparta en las
comunidades con tradiciones lingüísticas propias será bilingüe.
                          TíTULO II
                DE LOS DERECHOS, LAS GARANTíAS
                        Y LOS DEBERES
                          CAPíTULO 1
                DE LOS DERECHOS FUNDAMENTALES
ARTíCULO 11. El derecho a la vida es inviolable. No habrá
pena de muerte.
ARTíCULO 12. Nadie será sometido a desaparición forzada, a
torturas ni a tratos o penas crueles, inhumanos o degradantes.
ARTíCULO 13. Todas las personas nacen libres e iguales ante
la ley, recibirán la misma protección y trato de las autoridades y
gozarán de los mismos derechos, libertades y oportunidades sin
ninguna discriminación por razones de sexo, raza, origen na-
cional o familiar, lengua, religión, opinión política o filosófica.
El Estado promoverá las condiciones para que la igualdad sea
real y efectiva y adoptará medidas en favor de grupos discrimi-
nados o marginados.
El Estado protegerá especialmente a aquellas personas que por
su condición económica, física o mental, se encuentren en cir-
                          [ 15 ]
```

La página 15 del documento PDF original se muestra en la figura 4.6. Como puede observarse, salvo lo relacionado con los formatos, tipos de letras y colores, el texto obtenido a través de la función `pdf_text` corresponde línea por línea en su totalidad con el archivo original.

Figura 4.6. Ejemplo de archivo PDF



Constitución Política de Colombia 1991

De igual manera, la política exterior de Colombia se orientará hacia la integración latinoamericana y del Caribe.

ARTÍCULO 10. El castellano es el idioma oficial de Colombia. Las lenguas y dialectos de los grupos étnicos son también oficiales en sus territorios. La enseñanza que se imparta en las comunidades con tradiciones lingüísticas propias será bilingüe.

TÍTULO II
DE LOS DERECHOS, LAS GARANTÍAS Y LOS DEBERES

CAPÍTULO 1
DE LOS DERECHOS FUNDAMENTALES

ARTÍCULO 11. El derecho a la vida es inviolable. No habrá pena de muerte.

ARTÍCULO 12. Nadie será sometido a desaparición forzada, a torturas ni a tratos o penas crueles, inhumanos o degradantes.

ARTÍCULO 13. Todas las personas nacen libres e iguales ante la ley, recibirán la misma protección y trato de las autoridades y gozarán de los mismos derechos, libertades y oportunidades sin ninguna discriminación por razones de sexo, raza, origen nacional o familiar, lengua, religión, opinión política o filosófica.

El Estado promoverá las condiciones para que la igualdad sea real y efectiva y adoptará medidas en favor de grupos discriminados o marginados.

El Estado protegerá especialmente a aquellas personas que por su condición económica, física o mental, se encuentren en cir-

[15]

Fuente: Tomado de la Constitución Politica de Colombia

Una vez se tenga el documento PDF representado como una cadena de caracteres es posible utilizar las diferentes funciones de procesamiento de texto de la plataforma R, algunas de ellas mostradas en la siguiente sección. Para finalizar lo concerniente a documentos en formato PDF, se invita al lector a consultar otras funciones disponibles en la librería, en particular aquellas relacionadas con la generación de un documento PDF.

4.2. Procesamiento de cadenas de caracteres

El procesamiento de los archivos PDF o del texto extraído de las imágenes requiere la manipulación y el procesamiento de cadenas de caracteres, por ello R viene provisto de buen número de funciones para procesar este tipo de datos. Algunas de las funciones más utilizadas tienen que ver con la concatenación de cadenas de caracteres, la extracción de parte de una cadena o quizás las más interesantes, aquellas

relacionadas con buscar cadenas de caracteres que cumplan algún patrón. Este patrón se especifica normalmente como una *expresión regular*, metalenguaje para procesamiento de cadenas soportado no solo por R, sino por la gran mayoría de lenguajes de programación e incluso como herramientas en sí mismas (por ejemplo, `egrep`).

4.2.1. Concatenación de cadenas de caracteres

Para concatenar cadenas de caracteres, R provee la función `cat`., que recibe por parámetro un arreglo de cadenas de caracteres y retorna la concatenación de cada cadena involucrada en el arreglo separadas por un espacio. Esto se muestra en el siguiente ejemplo:

```
# Concatenación mediante cat
nombre <- c("José", "Nelson", "Pérez")
cat(nombre)
José Nelson Pérez>
```

Una de las desventajas de esta forma de utilización es que no es posible asignar el resultado de la concatenación a un objeto. Esta es la razón por la cual para concatenar cadenas de caracteres se utiliza en la práctica la función `paste`, la cual es un poco más general que la función `cat`. La función `paste` recibe como argumento todas las variables que quieran concatenarse como cadena de caracteres y se utiliza según se muestra a continuación:

```
# Concatenación mediante paste
nombre_completo <- paste( "José", "Nelson", "Pérez" )
nombre_completo
[1] "José Nelson Pérez"
```

Por defecto, la función `paste` utiliza un espacio como caracter separador, comportamiento que puede cambiarse utilizando el argumento `sep`.

```
# Concatenación mediante paste con separador no estándar
nombre_completo <- paste( "José", "Nelson", "Pérez", sep="" )
nombre_completo
[1] "JoséNelsonPérez"
nombre_completo <- paste( "José", "Nelson", "Pérez", sep="#" )
nombre_completo
[1] "José#Nelson#Pérez"
```

El proceso inverso de concatenación de una cadena es dividirla dando un caracter separador. Para realizar este proceso se utiliza la función `strsplit`, que recibe por parámetro la cadena a separar y el caracter separador. Supóngase que se quiere separar una cadena de texto para obtener todas las palabras del mismo de forma

separada. En este caso, el separador es el caracter espacio (“ ”) y se utiliza la función como se muestra en el siguiente ejemplo.

```
# Separación de una cadena en varias palabras
nombre <- "José Nelson Pérez"
strsplit( nombre, " ")
[ [1] ]
[1] "José" "Nelson" "Pérez"
```

La función `strsplit` devuelve por resultado una lista, estructura que en ocasiones puede ser inconveniente. Si se desea obtener las distintas separaciones del texto como un objeto de tipo vector se debe utilizar la función `unlist`, según se muestra en el siguiente ejemplo.

```
# Separación de una cadena en varias palabras
# como un vector
nombre_completo <- unlist(strsplit( nombre, " "))
nombre_completo
[1] "José" "Nelson" "Pérez"
nombre_completo[1]
[1] "José"
nombre_completo[2]
[1] "Nelson"
nombre_completo[3]
[1] "Pérez"
```

4.2.2. Extracción de subcadenas y búsqueda por expresiones regulares

Una funcionalidad muy común a la hora de procesar cadenas de caracteres es la capacidad para extraer una sección de caracteres de una variable. Para este fin, R provee la función `substring`, que recibe como parámetro un objeto R de tipo cadena de caracteres, la posición del primer caracter y la posición del último caracter de la subcadena. Si esta última posición no se hace explícita como argumento se extrae la subcadena a partir de la posición inicial (que siempre debe suministrarse).

```
# Subcadenas
vuelo <- "AV112"
aerolinea <- substring( vuelo, 1, 2)
numero_vuelo <- substring( vuelo, 3 )
aerolinea
[1] "AV"
numero_vuelo
[1] "112"
```

Otra funcionalidad aplicada a la hora de realizar procesamiento de texto es la capacidad para hacer una indagación en un texto utilizando un término de búsqueda. R provee esta aplicabilidad mediante la función `grep`, que en su versión más básica recibe por parámetro una *expresión regular* y una variable con tipo de dato alfanumérico y puede retornar por resultado todos aquellos índices de las observaciones que cumplen con el criterio de búsqueda o las coincidencias resultantes de la búsqueda como tal.

Que se retornen las coincidencias o el índice de las mismas depende de un argumento denominado `value` en la invocación de la función `grep`. Si se invoca con `value=T`, la función retorna las coincidencias de la búsqueda, mientras si se invoca `value=F`, el cual es el comportamiento por defecto de la función, se retornan los índices de las observaciones que coinciden con la exploración. Si adicionalmente se desea que el criterio de búsqueda sea independiente de si el texto está en mayúsculas o minúsculas se utiliza el parámetro adicional `ignore.case` con un valor de "verdadero" (`T`) en el llamado de la función.

```
resultados <- grep("expresion_regular", dataset$variable,
                   value = T, ignore.case=T)
```

Ahora bien, surgen las preguntas: ¿qué es una expresión regular, para qué sirve y cómo se especifica? Una expresión regular se define como un patrón que describe un conjunto de cadenas de caracteres con una estructura común y se especifica como expresiones en un *metalenguaje* con una sintaxis particular; este metalenguaje es diferente al lenguaje utilizado en el texto que se quiere consultar o cuyos patrones se quieren buscar. En la práctica, las expresiones regulares son utilizadas para buscar coincidencias de patrones en un texto o para la operación *buscar/reemplazar* común en algunos editores de texto.

Si bien este metalenguaje utilizado en las expresiones regulares se encuentra estandarizado en el código POSIX 1003.2, en la práctica existen dos versiones del metalenguaje, que aunque tienen muchas cosas en común, también tienen diferencias que pueden llegar a ser sustanciales en algunas aplicaciones. La versión del metalenguaje descrita en el estándar POSIX 1003.2 y utilizada en este libro, se conoce como *expresiones regulares extendidas*; alternativamente, existe una versión del metalenguaje conocido como *estilo PERL*, ya que es el usado en el lenguaje de programación PERL y en numerosas utilidades encontradas en sistemas operativos tipo UNIX. Por fortuna, R soporta ambos tipos de expresiones, que se utilizarán en este libro.

Cuando se habla de un metalenguaje, se hace referencia a una *nueva sintaxis* que debe ser utilizada para especificar la expresión regular. Dentro de esta sintaxis se cuenta con *metacaracteres*, esto es, símbolos de este metalenguaje que tienen un

significado específico a la hora de definir la expresión regular, pero que pueden tener un significado diferente fuera de este lenguaje. En las expresiones regulares extendidas, los metacaracteres son los siguientes:

```
$ * + . ? [ ^ ] | ( \
```

Se insiste en que estos metacaracteres tienen un significado diferente y específico dentro de la expresión regular, el cual difiere al significado tradicional que pueda tener cada uno de los símbolos.

Una introducción completa a las expresiones regulares puede ampliarse consultando a [8], pero en este libro se hará énfasis en dos de los metacaracteres: `^` y `$`.

El metacaracter `^` se interpreta como "al comienzo" y es utilizado cuando se desea obtener concordancias que comiencen por un prefijo determinado. En este caso la expresión regular consta del metacaracter seguido del prefijo particular.

```
# Expresión regular para prefijos
resultados <- grep("^prefijo", dataset$variable,
                   value = T, ignore.case=T)
```

El metacaracter `$` se interpreta como "al final" y es utilizado cuando se desea obtener concordancias que terminen con un sufijo determinado. En este caso la expresión regular consta del metacaracter seguido del sufijo particular.

```
# Expresión regular para sufijos
resultados <- grep("sufijo$", dataset$variable,
                   value = T, ignore.case=T)
```

Es válido ingresar una expresión regular que no contenga ningún metacaracter. En este caso se requiere únicamente un patrón que representa la cadena frente a la cual se evalúan las concordancias. Un filtro por una expresión regular de este tipo retorna todas aquellas concordancias que contengan el patrón, sin importar si este se encuentra al comienzo o al final del texto.

```
# Expresión regular para filtro
resultados <- grep("patron", dataset$variable,
                   value = T, ignore.case=T)
```

La referencia [8] es una excelente fuente de información sobre todo lo relacionado con expresiones regulares. Se invita al lector a consultar este texto y sus enlaces en internet para no solo obtener destreza en el diseño de expresiones regulares, sino también para apreciar la gran potencialidad de este tipo de herramientas.

4.2.3. Otras funciones de manipulación de cadenas de caracteres

Existen otras funcionalidades comunes a la hora de manipular cadenas de caracteres, como convertir una cadena a mayúsculas/minúsculas o reemplazar algunos de los caracteres. Para convertir una cadena de caracteres a mayúsculas se utiliza la función `toupper`, mientras la función `tolower` convierte una cadena a minúsculas.

```
nombre <- "José Nelson Pérez"
# Conversión a mayúsculas
toupper(nombre)
[1] "JOSÉ NELSON PÉREZ"
# Conversión a minúsculas
tolower(nombre)
[1] "josé nelson pérez"
```

Para sustituir parte del texto en una cadena de caracteres se utiliza la función `sub`, la cual recibe por argumento una *expresión regular* que representa el patrón que debe ser reemplazado en el texto, la nueva cadena con la cual va a ser cambiada la primera ocurrencia encontrada del patrón y la cadena de caracteres en la cual se realizará el proceso de reemplazo. Si en lugar de reemplazar la primera ocurrencia se desean sustituir todas las ocurrencias del patrón se debe utilizar la función `gsub`, cuyos argumentos son idénticos para fines prácticos. A manera de ejemplo, se desea reemplazar el texto "Nelson" por "Nelsinho"; para ello se utilizará la función `sub`, como se muestra a continuación:

```
nombre_completo
[1] "José"    "Nelson"    "Pérez"
# Reemplazo de texto
sub( "son$","sinho",nombre_completo[2] )
[1] "Nelsinho"
```

El ejemplo inmediatamente anterior puede interpretarse como sigue: se utilizó la función `sub` para reemplazar en el texto "Nelson" la coincidencia "son" por el nuevo patrón "sinho". La expresión regular "son$" tiene en cuenta el significado del metacarácter "$", que indica que debe reemplazarse el texto que *finaliza* con el sufijo "son". Cualquier expresión regular válida puede utilizarse como patrón de reemplazo en las funciones `sub` o `gsub`.

4.3. Bases de datos relacionales

A medida que se empieza a trabajar con conjuntos de datos más grandes, los archivos .CSV o los data frames de R empiezan a presentar algunos problemas, principalmente

porque en R todos los archivos o data frames deben estar en la memoria principal para su procesamiento. Cuando el tamaño del conjunto de datos es superior al tamaño de la memoria principal, el sistema operativo empieza a hacer uso de estrategias como memoria virtual o memoria *swap* (de intercambio), lo cual genera problemas de desempeño en algunos análisis intensivos en operaciones de memoria.

En este contexto adquiere valor manipular los datos directamente sobre una base de datos (generalmente de tipo relacional) y "traer" los datos a R únicamente cuando es necesario a través de una operación de consulta del tipo `SELECT` utilizando el lenguaje estándar SQL; al hacer esto se aprovecha la memoria principal restante para los cálculos involucrados en los análisis. Adicionalmente, y con el advenimiento del paradigma de computación en la nube, una base de datos relacional se presenta como una alternativa más segura que los archivos planos, en algunos casos con posibilidades de replicación y de respaldo de datos a muy bajo precio. Si a esto se le suma el hecho de poder escalar a medida que se requiere más capacidad o poder de procesamiento, es claro por qué algunas aplicaciones mantienen sus conjuntos de datos en una base de datos en lugar de exportarlos como un archivo plano en formatos como .CSV.

Si bien la variedad de manejadores de bases de datos relacionales es amplia, siendo populares alternativas como MySQL, MariaDB, SQL Server, PostgresQL, Oracle Database o MS Access, las funcionalidades para acceder a cualquiera de ellas son similares. Las principales funciones que proveen las librerías R para acceder a una base de datos relacional son las siguientes:

- Establecer una conexión a la base de datos.
- Leer una tabla de una base de datos y guardarla en un data frame.
- Ejecutar una consulta en la base de datos y obtener los resultados.
- Escribir o actualizar un data frame a una tabla en una base de datos.
- Eliminar una base de datos.
- Cerrar la conexión a la base de datos.

Sin embargo, existen particularidades en cada uno de los manejadores de bases de datos que han llevado a la implementación de librerías R especializadas en cada tecnología, las cuales son el tema de la siguiente sección.

4.3.1. Acceso a bases de datos utilizando R

Dependiendo del manejador de bases de datos particular, R provee paquetes especializados (también denominadas como interfaces en algunas referencias) que permiten la comunicación y manipulación de los diversos elementos que se encuentran en las diferentes bases de datos. Algunas de las librerías R utilizadas para este fin son:

- `RODBC`, utilizada para acceder a bases de datos a través del estándar *Open DataBase Connectivity* o Conectividad Abierta a Bases de Datos (ODBC) [12]. Ejemplos de bases de datos que utilizan esta interfaz son las basadas en tecnologías Microsoft como Access y SQL Server.
- `RMySQL`, que provee una interfaz para acceder a bases de datos MySQL [9].
- `RMariaDB`, actualización/reemplazo de la librería `RMySQL` para acceder a bases de datos MySQL/MariaDB [10].
- `ROracle`, que provee una interfaz para acceder a la base de datos del fabricante Oracle [13].
- `RJDBC`, paquete para acceder a bases de datos a través de una interfaz *Java DataBase Connectivity* o Conectividad Java a Bases de Datos (JDBC) [11].

Dado que hacer énfasis en cada una de estas librerías está fuera de los objetivos del libro, esta sección se centrará en la funcionalidad provista por la librería `RMariaDB`, resaltando que funcionalidades similares o análogas de seguro se encuentran en las otras interfaces. Las referencias al final del capítulo enlazan a la documentación de cada librería, junto con ejemplos de utilización de la misma.

Para que el paquete `RMariaDB` esté disponible en el espacio de trabajo de R, esta debe ser instalada e importarse con la directiva `library(RMariaDB)`.

```
# Instalación del paquete RMariaDB

install.packages("RMariaDB")
library(RMariaDB)
library(DBI)
```

El primer paso para obtener o guardar información en una base de datos es establecer una conexión con esta. Para ello, es necesario conocer (al menos) la siguiente información: nombre de usuario de la base de datos, contraseña de ese usuario, el nombre o la dirección IP de la máquina en la cual se encuentra el sistema gestor de bases de datos y el nombre de la base de datos, ya que en general un sistema gestor de bases de datos puede almacenar multitud de estas, cada una con varias tablas. Con esta información, es posible utilizar la función `dbConnect` con los parámetros `user`, `password`, `host` y `dbname` para identificar el nombre de usuario, la contraseña, el nombre o dirección del host y el nombre de la base de datos, respectivamente.

A manera de ejemplo, se desea establecer conexión con una base de datos MySQL/MariaDB llamada `"Personas"`, ubicada en `"db.introdatascience.org"`. El usuario para acceder a este sistema MySQL/MariaDB es `nperez`,

cuya contraseña es a1b2c3d4. Hay que aclarar que los respectivos permisos para acceder al sistema gestor con este usuario deben haber sido configurados previamente por el administrador del sistema gestor de base de datos. El código R para establecer esa conexión es el siguiente:

```
# Conexión a una base de datos MariaDB
conexion <- dbConnect( RMariaDB: :MariaDB(), user="nperez",
          password="a1b2c3d4", dbname="Personas",
          host="db.introdatascience.org" )
```

Existe un problema con la anterior forma de conectarse, y es que se están haciendo públicas las credenciales de acceso a la base de datos, situación no deseable. Para solventar este problema la solución propuesta por los desarrolladores de `RMariaDB` es crear un *grupo* en un archivo denominado ".my.cnf", ubicado en el directorio de trabajo de la sesión de R. Cuando se crea un grupo, los datos de conexión permanecen aglutinados y no compartidos. En el caso de `RMariaDB` la definición de un grupo en el archivo ".my.cnf" luce de la siguiente forma:

```
[rs-dbi]
database=Personas
user=nperez
password=a1b2c3d4
```

Naturalmente, el nombre de la base de datos y los datos particulares de conexión de cada usuario cambian dependiendo del caso. Una vez se define un grupo, la conexión se sigue realizando mediante la función `dbConnect`, pero utilizando el nombre del grupo como parámetro.

```
# Conexión a una base de datos MariaDB
# Credenciales no públicas
conexion <- dbConnect(RMariaDB: :MariaDB(), group="datascience" )
```

Una vez se cuenta con la conexión a la base de datos, es posible utilizarla para realizar diversas funciones, como listar las tablas de la base de datos, escribir un objeto R de tipo data frame como una tabla en la base de datos, leer una tabla completa y guardarla como un objeto data frame o enviar directamente consultas en el lenguaje SQL a la base de datos.

La función `dbListTables` es la encargada de listar todas las tablas que se encuentran en una base de datos y recibe como parámetro la conexión previamente establecida al recurso.

```
# Listar tablas de una base de datos
dbListTables( conexion )
```

La función `dbWriteTable` es la encargada de escribir un data frame en una tabla de la base de datos. Recibe por parámetro la conexión previamente establecida, el nombre de la nueva tabla y el data frame a salvar.

```
# Escribir un data frame como una tabla en la
# base de datos
dbWriteTable( conexion, "Personas", personas )
```

Si la tabla ya existe y se desea sobreescribir, debe adicionarse el argumento `overwrite` con el valor `TRUE`. Sin este argumento el llamado a la función generará un mensaje de error.

```
# Escribir un data frame como una tabla en la
# base de datos
dbWriteTable( conexion, "Personas", personas, overwrite=TRUE )
```

La función `dbReadTable` realiza el proceso inverso: guarda una tabla en un objeto de tipo data frame. La función recibe por argumento la conexión establecida con la base de datos y el nombre de la tabla que se desea guardar como tal.

```
# Leer una tabla como un data frame
personas <- dbReadTable( conexion, "Personas" )
```

En `RMariaDB` existen dos formas para enviar y procesar una consulta SQL. La primera es haciendo uso de la función `dbGetQuery`, que recibe la conexión a la base de datos y la consulta SQL y retorna directamente un objeto de tipo data frame.

```
# Envío de una consulta SQL
resultados <- dbGetQuery( conexion, "SELECT * FROM Personas")
```

La segunda forma es haciendo uso de las funciones `dbSendQuery` y `dbFetch`. La función `dbSendQuery` es similar a `dbGetQuery`, pero en lugar de dar por resultado un objeto de tipo data frame retorna un objeto de una naturaleza diferente perteneciente a la clase `MariaDBResult`. Para obtener la información de este objeto de tipo `MariaDBResult` se utiliza la función `dbFetch`, la cual recibe por parámetro el objeto de tipo `MariaDBResult` y un número entero con el número de resultados que se desean obtener del total disponible. La función `dbFetch` retorna un data frame.

```
# Envío de una consulta SQL.
# Uso de dbSendQuery y dbFetch
res <- dbSendQuery( conexion, "SELECT * FROM Personas")
df <- dbFetch( res, n=10 ) # 10 resultados por invocación
```

Es posible realizar una limpieza de los resultados almacenados en un objeto de tipo `MariaDBResult`. Para ello se utiliza la función `dbClearResult`.

```
# Limpieza de resultados
dbClearResult( res )
```

Otras funciones importantes tienen que ver con la eliminación y verificación de existencia de una tabla en la base de datos. Para eliminar una tabla se utiliza la función `dbRemoveTable`, que la elimina dada una conexión con el gestor de bases de datos y el nombre de la tabla. Para verificar la existencia de una tabla se utiliza la función `dbExistsTable` con idénticos argumentos. Esta función retorna un valor lógico `TRUE` en caso de que la tabla exista o `FALSE` en caso contrario.

```
# Eliminación de una tabla
dbRemoveTable( conexion, "Personas" )
# Verificación de existencia de un tabla
dbExistsTable( conexion, "Personas" )
```

Finalmente, para cerrar la conexión a la base de datos se utiliza la función `dbDisconnect`. Cerrar la conexión es una práctica recomendada siempre que se trabaje con sistemas gestores de bases de datos.

```
# Cerrar la conexión a la base de datos
dbDisconnect(conexion)
```

4.4. Fuentes sindicadas

El término *sindicación* o redifusión tiene que ver con la distribución de contenido informativo (creado por un emisor) por parte de un segundo actor a través de una licencia o contrato. El término es común en la terminología de los medios de comunicación como radio, prensa y televisión, por ejemplo, los derechos de retransmisión o repetición de una serie.

Sin embargo, dentro del contexto de los contenidos web como fuentes de datos, se entiende por una *fuente sindicada*, canal o fuente web (*web feed*) a aquel medio de difusión de contenido web que coloca información como documentos a disposición de un tercero, que puede ser un suscriptor individual u otra fuente, que también lo puede

compartir. En lugar de un contrato para la redifusión, suelen haber unas condiciones de uso con los derechos de redistribución, usualmente gratuitos. Son mayormente utilizadas en la prensa electrónica para suministrar novedades y noticias actualizadas, aunque también es común encontrar este tipo de fuente en sitios de *blogs* y cada vez más en otras aplicaciones como comercio en línea.

Los dos principales formatos utilizados por las fuentes sindicadas son Redifusión Realmente Simple o *Really Simple Syndication* (RSS) y Atom. Ambos están basados en XML, que constan principalmente de un título, una fecha de publicación de la información, un resumen del contenido y un enlace web con la información ampliada, más algunos metadatos como el autor. En general, se puede afirmar que son formatos diseñados para ser entendible más fácilmente por máquinas que por humanos, si bien existen aplicaciones conocidas como *agregadores* o "lectores RSS", similares a un lector de correo electrónico, que permiten suscribirse a varias fuentes y obtener su contenido de manera más amigable para los humanos.

Utilizar fuentes sindicadas tiene muchos beneficios como evitar publicidad, virus y *spam*, pero la principal ventaja es el ahorro de tiempo, ya que es posible acceder a contenidos nuevos de interés de múltiples sitios sin tener que visitarlos uno por uno. Adicionalmente, las fuentes sindicadas permiten ser consultadas sin intervención humana a través de "robots", programas especializados encargados de automatizar el proceso de consulta.

4.4.1. Fuentes sindicadas en R

Para la consulta de fuentes sindicadas o *feeds*, R provee una extensión llamada `FeedeR` [14]. `FeedeR` permite leer de la fuente tanto en formato RSS como en formato Atom, utilizando para ello una misma función denominada `feed.extract`, que recibe por parámetro la URL de la fuente. La instalación del paquete se realiza o bien de la forma estándar `(install.packages("feedeR")` o a través de la herramienta `devtools`, utilizada para facilitar la administración de paquetes en R. Como en todos los casos, para que el paquete esté disponible en el espacio de trabajo de R, debe importarse con la directiva `library(feedeR)`.

```
# Instalación del paquete feedeR
install.packages("feedeR")
library(feedeR)
```

Con `feedeR` instalado, se utiliza la función `feed.extract`. Por ejemplo, se quiere leer un artículo del canal web de la sección "Tecnología" del periódico colombiano "El Tiempo" (https://www.eltiempo.com), disponible en el Localizador Uniforme de Recursos (URL) "http://www.eltiempo.com/rss/tecnosfera.xml". El código para lograr este objetivo se muestra a continuación:

```
eltiempo_tecnologia <-
  feed.extract("http://www.eltiempo.com/rss/tecnosfera.xml")
```

La función `feed.extract` retorna un objeto de tipo lista (ver sección 3.2.5.) con cuatro componentes: tres componentes describen los metadatos del contenido y el otro atributo es la información propiamente dicha, el cual es un objeto de tipo data frame. Los atributos que describen los metadatos son: `title`, que representa el título del contenido, `link` que consigna el enlace URL en donde está el contenido y `updated`, que hace referencia a la fecha de actualización del contenido, el cual no siempre se encuentra definido.

```
eltiempo_tecnologia$title
[1] "EL TIEMPO.COM - TecnologÃa"
eltiempo_tecnologia$link
[1] "http://www.eltiempo.com/rss/tecnosfera.xml"
eltiempo_tecnologia$updated
[1] NA
```

El componente de la lista con la información propiamente dicha se denomina `items`. Este es un objeto de tipo data frame con las siguientes variables: `title` con el título de la entrada, `date` con la fecha de la entrada y `link` con el enlace donde se encuentra el contenido web con la información completa del artículo. Adicionalmente, la tabla de datos cuenta con una variable `hash` con información de verificación de integridad de la entrada.

```
eltiempo_tecnologia$items[ 15, c("title","date") ]
  title   date
15 Archivan iniciativa ciudadana que buscaba legalizar
  Uber en Colombia 2017-04-19 16:25:07
eltiempo_tecnologia$items[ 13,c("link","hash") ]
  link    hash
13 http://www.eltiempo.com/tecnosfera/novedades-
tecnologia/xiaomi-mi-6-el-iphone-7-chino-79458 b82e7e933e...
```

A la fecha de escritura de este libro el paquete `feedeR` se encuentra todavía en sus versiones iniciales, razón por la cual no decodifica correctamente la información de algunos canales web. La anterior afirmación es especialmente cierta para algunos canales web en idioma español, razón por la cual es posible que, si bien la información de una fuente sindicada puede ser leída desde un programa agregador, no va a ser posible cargarla en el entorno de trabajo de R. Posiblemente en futuras versiones del paquete, la compatibilidad con un número mayor de canales web aumentará.

Si bien una fuente sindicada provee información de interés en temas particulares, el contenido completo de las entradas o *feeds* todavía se encuentra en algún lugar de la web que únicamente está referenciado por la variable `link` del data frame `items`. ¿Cómo se hace entonces para acceder a esa información y eventualmente analizar el texto y las imágenes en ella contenidas? Ese es el tema de la siguiente sección, *web scraping*.

4.5. Web scraping

Por *web scraping* se entiende el proceso automático de recolectar datos desde sitios y páginas web. Los datos en la red global son abundantes, pero en ocasiones no se encuentran en formatos listos para ser procesados, como .CSV o `JSON`, sino que están "incrustados" en alguna página o sitio web, algunas de ellas dinámicas, en formatos cuya estructura puede ser más compleja.

Quizás el caso más sencillo para obtener información en un proceso de *web scraping* es aquel en el que el conjunto de datos se encuentra en un archivo público y es directamente accesible en un sitio web cuya dirección, nombre o en general su URL es conocida. En este caso se puede hacer uso directo de la función `read.table`, que recibe como parámetro la dirección completa del recurso (archivo), incluyendo el protocolo de acceso y la extensión del archivo. Debe notarse que si el protocolo del acceso al archivo requiere alguna autenticación previa, como FTP, no puede utilizarse esta función.

```
# Lectura de un recurso web mediante URL
archivo <-
  read.table("http://direccion-sitio/archivo.extension")
```

En general los conjuntos de datos no suelen estar disponibles directamente para descarga. Debido a que los datos se encuentran distribuidos por toda la red global, estos deben estar conformes con las principales tecnologías utilizadas en la red. Los pilares tecnológicos de los sitios web, hoy en día son básicamente HTML/XML, AJAX y JSON. Estos fueron explicados previamente y son descritos así:

- HMTL. El Lenguaje de Marcado de Hipertexto o *Hypertext Markup Language* es el lenguaje estándar utilizado por los navegadores de internet para estructurar y visualizar el contenido web. Si bien HTML no es un formato de almacenamiento en sí mismo, frecuentemente contiene, sobre todo en aquellas páginas web estáticas, la información de interés; esta puede estar "camuflada" en tablas, listas, enlaces u otras estructuras del lenguaje. Sin embargo, la forma como realmente se encuentran los datos en la página web con código HTML difiere de la forma como el navegador visualiza los datos, razón por la cual es necesario

procesar la página web y de alguna forma decodificar las estructuras de lenguaje HTML para acceder a la información. Como es de esperarse, R ya posee objetos encargados de esta funcionalidad que se encargan de este procesamiento sin que quien use la función sea un experto.

- XML. El Lenguaje de Marcado Extensible o *EXtensible Markup Language* es uno de los formatos más populares para intercambiar datos en la web. Está estrechamente relacionado con HTML, pues ambos son lenguajes de marcado o basados en etiquetas. La diferencia principal radica en sus objetivos: HTML está principalmente pensado para visualización de la página web en el navegador, mientras XML tiene como función almacenar los datos. De forma similar a HTML, la plataforma R ya provee objetos encargados de entender este lenguaje, interpretar sus estructuras y obtener la información en otro formato más manejable, como lo puede ser un data frame.
- AJAX. Conjunto de tecnologías que permite a los sitios web solicitar datos de forma asincrónica en una sesión, a la vez que actualiza su apariencia visual en el navegador de manera dinámica. Desde el punto de vista de los *web scrappers* estas tecnologías constituyen un obstáculo para la obtención del contenido del sitio, pues se apartan un poco del enfoque HTML/HTTP. Sin embargo, aún con estas limitaciones es posible aprovecharse de ciertas funcionalidades incorporadas en los navegadores modernos para tener acceso al contenido, funcionalidades que si bien no son provistas de primera mano por R, son utilizables por la plataforma.

El objetivo de un proceso de *scrapping* es lograr de alguna forma representar el contenido de un documento HTML/XML o cualquiera de las otras tecnologías como un objeto en una sesión de R; una vez se cuente con la representación del documento en un objeto R es más sencillo realizar el proceso de extraer la información. Este proceso de importar archivos HTML o de otros formatos a una sesión R e interpretar su contenido se conoce como *parsing*, término que podría traducirse como *análisis gramático*. Sin entrar en los detalles de este proceso, la idea es utilizar un procedimiento que *entienda e interprete* la estructura de un documento, en este caso HTML, en lugar de únicamente imprimir el código de la página.

A manera de ejemplo, supóngase que se desea obtener la información de población de los países y territorios del mundo. Esta consulta se encuentra en Wikipedia, donde el lector podrá reconocer que la información solicitada se visualiza en una tabla junto con textos aclaratorios. Si no se utilizara un proceso de análisis léxico, es probable que únicamente se obtuviesen apartes del *código fuente* de la página, lo cual no es lo que se desea.

```
# Carga de un documento HTML sin análisis léxico
url <- "https://es.wikipedia.org/wiki/Anexo:Países_y_
                territorios_dependientes_por_población"
contenido <- readLines( con = url )

contenido[560:580]
[1] "<td align=\"center\">C</td>"
[2] "<td align=\"left\"><a rel=\"nofollow\" class=\"ext...
href=\"http://www.citypopulation.de/KoreaSouth-Mun.html...
[3] "</tr>"
[4] "<tr>"
[5] "<td>28</td>"
[6] "<td align=\"left\"><span class=\"flagicon\"><img...
src=\"//upload.wikimedia.org/wikipedia/commons/thumb/...
20px-Flag_of_Colombia.svg.png\" width=\"20\" height=\...
srcset=\"//upload.wikimedia.org/wikipedia/commons/thu...
vg/30px-Flag_of_Colombia.svg.png 1.5x,
...
[7] "<td align=\"left\">AmÃ©rica</td>"
[8] "<td>49 639 000</td>"
[9] "<td>0,66</td>"
[10] "<td>1,17</td>"
[11] "<td>579 000</td>"
[12] "<td>0,56</td>"
[13] "<td>60</td>"
[14] "<td>49 443 000</td>"
...
```

Para la interpretación de contenido HTML/XML es necesario reconstruir la jerarquía de sus distintas etiquetas y entender esta representación, conocida como Modelo de Objetos de Documento o *Document Object Model* (DOM). También es posible cambiar esta estructura adicionando, eliminado o modificando los distintos elementos de la página HTML. Como es de esperarse, R ya provee funciones para este fin, luego, el objetivo es determinar cuáles funciones de R realizan estas tareas y su forma de utilización.

Una de la principales funciones provistas por R para el procesamiento de contenido HTML/XML es la función `htmlParse`. Esta recibe como principal argumento un archivo o texto en formato HTML/XML y retorna un objeto con la estructura interna de la página en representación del modelo de objetos. Esta representación es de varios tipos, en particular `HTMLInternalDocument, XMLInternalDocument` y `XMLAbstractDocument.`

```
# Carga de un documento HTML con análisis
# léxico
url <- "https://es.wikipedia.org/wiki/
   Anexo:Países_y_territorios_dependientes_por_población"
contenido <- readLines( con = url )
territorios <- htmlParse( contenido )
class( territorios )
[1] "HTMLInternalDocument" "HTMLInternalDocument"
      "XMLInternalDocument" "XMLAbstractDocument"
```

Una vez se cuenta con una representación interna de un documento HTML/XML apropiada para su procesamiento, es posible invocar funciones R especializadas que hacen la labor de extracción del contenido de manera más sencilla. De otra forma (y en algunos problemas muy particulares) sería necesario procesar esta representación mediante expresiones regulares, procesamiento de cadenas de caracteres u otras tecnologías como XQuery.

Una de esas funciones es la `readHTMLTable`, encargada de obtener todas las tablas que se encuentren en el contenido HTML/XML. Esta función es responsable de buscar en la estructura del documento etiquetas HTML relacionadas con tablas (`<td>`,`<tr>`, etc...) y extraer la información de cada una de estas en formato data frame. La función retorna un listado de todas las tablas presentes en el documento HTML/XML, cada una de ellas como un elemento de una lista. En ocasiones y dependiendo de la complejidad y el diseño del contenido HTML/XML puede suceder que algunos de los elementos de la lista que retorna la función `readHTMLTable` no cuenten con contenido alguno, hecho representado en R como una definición del lenguaje denominado `NULL`.

```
# Lectura de tablas de un documento HTML
tablas <- readHTMLTables( territorios )
class( tablas )
[1] "list"
tablas[ [1] ]
NULL
tabla1 <- tablas[ [3] ]
class( tabla1 )
[1] "data.frame"
```

Al estar la información en formato data frame se pueden aplicar las técnicas expuesta en este y en el capítulo anterior para el procesamiento y análisis de la información. En ocasiones es necesario un proceso de limpieza de datos, como corregir

los nombres de las variables o remover algunas observaciones. La siguiente sección tendrá más que decir respecto a este proceso de preparación de datos.

```
tabla2 <- tablas[ [4] ]
names(tabla2)[1] <- "Continente, subcontinente o
                                región geográfica"
names(tabla2)[5] <- "Cambio medio absoluto anual"
```

4.6. Preparación de datos

En la práctica es inusual que el conjunto de datos esté completamente preparado para el análisis; el conjunto puede tener datos faltantes, datos con errores, fuera de rango, en diferente escala o inconsistentes. Por ello, es necesario una etapa de preprocesamiento o preparación de los datos (también llamada en algunas referencias *etapa de limpieza*) para corregir estos aspectos. Además, es factible que las variables deban ser sometidas a alguna transformación para facilitar su análisis o la construcción de modelos que las utilicen.

4.6.1. Ordenamiento y eliminación de observaciones duplicadas

Un primer preprocesamiento del conjunto de datos puede consistir en un reordenamiento de este, respecto de alguna o varias variables o en la eliminación de observaciones duplicadas. Para estos fines, R provee las funciones `order` y `unique` que pueden ser invocadas con varios argumentos de acuerdo con la necesidad particular.

Comúnmente se desea reordenar el conjunto de datos en referencia con los valores de una variable (columna). En este caso, el único argumento de la función `order` es la variable de ordenamiento. Por ejemplo, el código R para ordenar un conjunto de datos almacenado en un objeto `datos` respecto a la variable `var` del mismo conjunto es:

```
# Reordenar respecto a una variable
datos <- datos[ order(datos$var), ]
```

También puede ser necesario ordenar de forma *descendente* en lugar de ascendentemente. En este caso la función `order` se invoca con el argumento `decreasing=T`. El mismo ejemplo anterior ordenado de forma descendente respecto a la variable *var* se escribiría en R como:

```
# Reordenar respecto a una variable
# Orden descendente
datos <- datos[ order(datos$var, decreasing=T), ]
```

Igualmente, es posible reordenar el conjunto de datos por más de una variable. Por ejemplo, si se quiere ordenar un data frame almacenado en un objeto `personas` primero por su variable `edad` y luego por su variable `estatura`, debería invocarse la función `order` como se muestra a continuación:

```
# Reordenar respecto a varias variables
personas <- personas[ order(personas$edad, personas$estatura), ]
```

Para la eliminación de observaciones duplicadas en un conjunto de datos en R, se utiliza la función `unique`. Esta recibe como parámetro un data frame o conjunto de datos y devuelve como parámetro otro conjunto de datos en el cual ninguna observación se repite. Debe aclararse que R considera que dos observaciones están repetidas si tienen los mismos valores en *todas* las variables del conjunto.

```
# Eliminación de duplicados
datos <- unique(datos)
```

4.6.2. Datos aislados (Outliers)

Una primera inspección de una variable de respuesta o explicativa continua, posiblemente utilizando algún método de visualización como `plot`, puede dejar entrever que existen valores numéricos inusuales o alejados del resto de datos. Si bien estos valores, denominados *outliers*, no son necesariamente errores, sí es importante determinar la causa de su valor inusual, ya que puede ser consecuencia de un error en la captura del dato o un error en la escala.

Para una detección simple de estos valores, R provee una función denominada `which` que recibe como parámetro una condición que (se sospecha) cumple aquellos datos cuestionables y devuelve por resultado las posiciones de las observaciones que cumplen dicha condición. Esto con el objetivo de corregir el dato o analizar más en profundidad el por qué de su valor.

Supóngase que en un estudio clínico de personas adultas se encontraron en la variable `estatura`, valores como 155 o 0,45, los cuales pueden ser resultado de un error de digitación del dato en el primer caso o un valor que si bien es extraño puede deberse a una condición especial en el crecimiento del segundo caso. El analista de datos decidió que aquellos valores por fuera del intervalo [0, 50, 2, 40] pueden llegar a ser datos inusuales. La forma de identificar la posición en la variable se escribiría en R como:

```
# Detección de valores fuera de rango
which( estatura <= 0.50    |    estatura > 2.40 )
[1]   50   212
```

Se quiere recalcar en este punto la importancia de la visualización de las variables a la hora de identificar los datos aislados, así como enfatizar en que no todos los datos aislados son errores y pueden ser respuestas genuinas. Eso sí, hay que estar plenamente consciente de su existencia y sus causas a la hora de analizar el conjunto de datos.

4.6.3. Niveles en datos categóricos

Una forma de determinar si existen errores en una variable categórica es comparar el número de niveles de la variable en la tabla de datos contra el número esperado de niveles de la variable. Estos valores pueden diferir por errores de captura de datos (digitación) o por no tener en cuenta las mayúsculas/minúsculas en los nombres de los niveles, entre otras.

Para determinar si se presentan más niveles en una variable categórica de los esperados, R cuenta con dos funciones: `levels`, que lista los niveles de una variable y `table`, que muestra cuántas veces cada nivel aparece en una variable de una tabla de datos. Supóngase que se cuenta con una tabla de datos como se muestra a continuación:

```
          Estatura        peso   genero
1         1.68            55     F
2         1.84            91     M
3         1.78            88     m
4         1.55            47     F
5         1.89            105    M
```

En este caso, si se aplica la función `table` sobre la variable `genero`, se determina que se tienen tres niveles de la variable cuando lo esperado es tener únicamente dos. Esto es debido a que uno de los niveles fue digitado en letras minúsculas (“m”) en lugar de mayúsculas (“M”).

```
# Identificación de niveles inconsistentes
levels(personas$genero)
[1]   "F"   "m"   "M"
table(personas$genero)
F   m   M
2   1   2
```

Una vez se determinan aquellos niveles inconsistentes, se puede utilizar la función `which` para encontrar el número de observación y corregir la tabla de datos empleando la misma función `levels`. En la práctica, es común verificar variable por variable, sean estas continuas o categóricas para determinar errores como los expuestos hasta ahora.

```
which( personas$genero == "m" )
[1]   3
levels(personas$genero)  [3] <- "M"
```

Dado que en los modelos estadísticos que involucran variables categóricas, usualmente el primer nivel es utilizado como nivel de referencia, en ocasiones es necesario cambiar el orden de los niveles. Para lograr este cambio de orden, R provee la función `relevel`, a la cual se le especifica como argumento cuál debe ser el nivel que debe aparecer en primer lugar y por lo tanto sirve como nivel de referencia. Esta función se utiliza como se muestra a continuación.

```
# Cambio de orden de los niveles
levels(personas$genero)
[1]    "F"   "M"
personas$genero <- relevel( personas$genero, "M" )
levels(personas$genero)
[1]    "M"    "F"
```

4.6.4. Combinación de varios conjuntos de datos

En análisis de datos es común tener que realizar operaciones sobre observaciones que se encuentran almacenadas en dos o más conjuntos de datos. Esto puede involucrar la adición de nuevas variables al conjunto de datos, integrar dos conjuntos de datos con las mismas variables en uno solo o encontrar las observaciones en dos conjuntos de datos que tienen una variable en común.

La primera forma de combinación es la integración de dos conjuntos de datos (data frames) con *las mismas* variables, no necesariamente en el mismo orden. Esta igualdad implica que los nombres de las variables sean iguales. Si el conjunto de datos "data1" tiene la forma de la figura 2.1, y el conjunto de datos "data2" tiene la siguiente forma,

	x_1	x_2	x_3	$\cdots$	x_p
o_{n+1}	$x_{(n+1)1}$	$x_{(n+1)2}$	$x_{(n+1)3}$	$\cdots$	$x_{(n+1)p}$
o_{n+2}	$x_{(n+2)1}$	$x_{(n+2)2}$	$x_{(n+2)3}$	$\cdots$	$x_{(n+2)p}$
o_{n+3}	$x_{(n+3)1}$	$x_{(n+3)2}$	$x_{(n+3)3}$	$\cdots$	$x_{(n+3)p}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_{n+M}	$x_{(n+M)1}$	$x_{(n+M)2}$	$x_{(n+M)3}$	$\cdots$	$x_{(n+M)p}$

entonces el resultado de la integración por filas u observaciones es el siguiente:

	x_1	x_2	x_3	$\cdots$	x_p
o_1	x_{11}	x_{12}	x_{13}	$\cdots$	x_{1p}
o_2	x_{21}	x_{22}	x_{23}	$\cdots$	x_{2p}
o_3	x_{31}	x_{32}	x_{33}	$\cdots$	x_{3p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_n	x_{n1}	x_{n2}	x_{n3}	$\cdots$	x_{np}
o_{n+1}	$x_{(n+1)1}$	$x_{(n+1)2}$	$x_{(n+1)3}$	$\cdots$	$x_{(n+1)p}$
o_{n+2}	$x_{(n+2)1}$	$x_{(n+2)2}$	$x_{(n+2)3}$	$\cdots$	$x_{(n+2)p}$
o_{n+3}	$x_{(n+3)1}$	$x_{(n+3)2}$	$x_{(n+3)3}$	$\cdots$	$x_{(n+3)p}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_{n+M}	$x_{(n+M)1}$	$x_{(n+M)2}$	$x_{(n+M)3}$	$\cdots$	$x_{(n+M)p}$

Si se cumplen las anteriores condiciones, la función `rbind` permite que las observaciones de un conjunto de datos `data2` sean anexadas *al final* de un conjunto de datos `data1`.

```
# Anexar filas al final de un data frame
data.integrado <- rbind(data1, data2)
```

En esta *combinación por filas*, utilizando la función `rbind` no se identifican los duplicados ni se garantiza que los datos queden ordenados bajo algún criterio. También es posible anexar tres o más conjuntos de datos de una manera similar utilizando la función `rbind`.

```
# Anexar filas al final de un data frame
data.integrado <- rbind(data1, data2, data3)
```

Una segunda forma de combinación, es la integración de dos conjuntos de datos que contienen diferentes variables de las mismas observaciones, de forma que los conjuntos de datos quedan en un formato "uno al lado del otro". Si el conjunto de datos "data1" tiene la forma de la figura 2.1 y el conjunto de datos "data2" tiene la siguiente forma,

	x_{p+1}	x_{p+2}	x_{p+3}	$\cdots$	x_{p+K}
o_1	$x_{1(p+1)}$	$x_{1(p+2)}$	$x_{1(p+3)}$	$\cdots$	$x_{1(p+K)}$
o_2	$x_{2(p+1)}$	$x_{2(p+2)}$	$x_{2(p+3)}$	$\cdots$	$x_{2(p+K)}$
o_3	$x_{3(p+1)}$	$x_{3(p+2)}$	$x_{3(p+3)}$	$\cdots$	$x_{3(p+K)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_n	$x_{n(p+1)}$	$x_{n(p+2)}$	$x_{n(p+3)}$	$\cdots$	$x_{n(p+K)}$

entonces el resultado de la integración por columnas o variables es el siguiente:

	x_1	x_2	$\cdots$	x_p	x_{p+1}	x_{p+2}	$\cdots$	x_{p+K}
o_1	x_{11}	x_{12}	$\cdots$	x_{1p}	$x_{1(p+1)}$	$x_{1(p+2)}$	$\cdots$	$x_{1(p+K)}$
o_2	x_{21}	x_{22}	$\cdots$	x_{2p}	$x_{2(p+1)}$	$x_{2(p+2)}$	$\cdots$	$x_{2(p+K)}$
o_3	x_{31}	x_{32}	$\cdots$	x_{3p}	$x_{3(p+1)}$	$x_{3(p+2)}$	$\cdots$	$x_{3(p+K)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_n	x_{n1}	x_{n2}	$\cdots$	x_{np}	$x_{n(p+1)}$	$x_{n(p+2)}$	$\cdots$	$x_{n(p+K)}$

La función para lograr esto en R es `cbind`. Se puede afirmar que `cbind` es la función equivalente a `rbind`, solo que la primera opera a nivel de columnas y la segunda a nivel de filas (observaciones).

```
# Integración a nivel de columnas
data.integrado <- cbind(data1, data2)
```

De forma análoga a `rbind`, es posible combinar tres o más conjuntos de datos, siempre teniendo en cuenta que todos los conjuntos de datos deben tener el mismo número de filas (observaciones).

```
# Integración a nivel de columnas
data.integrado <- cbind(data1, data2, data3, data4)
```

Una tercera forma de combinar dos conjuntos de datos es realizando un "emparejamiento" de observaciones de acuerdo con variables que tienen en común ambos conjuntos de datos. En esta forma de combinación solo aquellas observaciones que aparezcan en ambos conjuntos de datos (de acuerdo con la(s) variable(s) en común) aparecerán en el resultado de la combinación. Aquellas variables no comunes también se anexarán en la combinación. La función utilizada en R para realizar esta función es `merge`, que recibe por parámetro los conjuntos de datos a integrar.

```
# Integración por variables comunes
data.integrado <- merge(data1, data2)
```

La función `merge` automáticamente identifica aquellas variables que comparten el mismo nombre y las utiliza para emparejar las observaciones. Puede suceder que si bien dos variables representen el mismo concepto no estén nombradas exactamente igual. En este caso es necesario especificar a la función `merge` cuáles son los nombres de las variables involucradas en la integración tanto para el primer conjunto de datos (argumento `by.x`) como para el segundo (argumento `by.y`).

```
# Integración por variables comunes
data.integrado <- merge(data1, data2, by.x="var1", by.y="VAR1")
```

Por último, si bien la función `merge` excluye aquellas observaciones que no fueron emparejadas por estar presentes solo en uno de los conjuntos de datos, es posible que sean incluidas si el análisis así lo requiere. Mayor información sobre esta funcionalidad y otras opciones de la función `merge` se encuentran en la documentación de la función.

4.6.5. Cambios en la estructura de los datos

En ocasiones es necesario realizar cambios mayores a la estructura de un conjunto de datos, ya que estos pueden encontrarse en un formato que, si bien corresponde a un data frame, puede ser inconveniente a la hora de manejar los datos organizados por grupos o subconjuntos. Considérese, a manera de ejemplo, el siguiente conjunto:

valor	categoria
9.21	Tipo I
7.53	Tipo I
7.48	Tipo II
8.08	Tipo III
11.51	Tipo II
12.79	Tipo III
10.15	Tipo III
11.85	Tipo I
9.42	Tipo II

Alternativamente, el anterior conjunto de datos puede presentarse de la siguiente forma:

Tipo I	Tipo II	Tipo III
9.21	7.48	8.08
7.531	8.08	12.79
11.85	11.51	10.15

Si bien esta segunda representación *no* está en formato de tabla de datos, pues los datos de cada fila no corresponden a la misma observación, puede en ocasiones ser un formato más conveniente para analizar los datos por múltiples criterios para resumir los datos agrupados en torno a una variable categórica o incluso si se desea modelar los valores en función de la variable de agrupamiento. La primera representación de los datos se conoce como *forma apilada*, mientras que la segunda se concibe como *forma no apilada*. Es de suponer que R provee funciones para convertir un conjunto de datos de forma apilada a no apilada y viceversa.

Para convertir de una forma no apilada a una apilada se utiliza la función `stack`.

```
# Conversión de forma no apilada a apilada
datos.apilados <- stack( datos.no.apilados )
```

La función `stack` automáticamente renombra las nuevas variables a los valores `values` e `ind`. Tambien es posible reasignar estos nombres a algo más informativo con la función `names`, explicada anteriormente.

```
names( datos.apilados ) <- c("valor", "categoria")
```

Si el conjunto de datos ya se encuentra en un formato de dos variables con los valores y las variables de agrupamiento (como el mostrado anteriormente), la función `unstack` convierte de un formato no apilado a un formato apilado, así:

```
# Conversión de forma apilada a no apilada
datos.no.apilados <- unstack( datos.apilados )
```

Sin embargo, no es muy común que el conjunto de datos se encuentre en este formato, por lo cual es necesario especificar a la función `unstack` cuál es la variable con los valores y cuál es la variable con los grupos. Para ello se utiliza un formato *var.valores ~ var.grupos*, de forma que la función `unstack` debe invocarse de la siguiente forma:

```
datos.no.apilados <- unstack( datos.apilados,
                                  var.valores~var.grupos )
```

Cabe mencionar, a manera de restricción, que para que este proceso de conversión de datos apilados a no apilados funcione según lo esperado, el número de valores en cada uno de los grupos debe ser el mismo.

4.6.6. Otras transformaciones

Existen otras operaciones que pueden ser realizadas sobre un conjunto de datos con el objetivo de facilitar su análisis. Una de ellas es la creación de una nueva variable categórica resultado de clasificar las observaciones de acuerdo con el valor de una variable continua. Este tipo de transformación debe ser sometida a amplio escrutinio, pues las variables continuas proveen más información que las variables categóricas, por lo cual es mejor utilizar en los modelos las variables en su forma continua.

Si en definitiva el análisis lo requiere, R provee la función `cut` para la creación de esta nueva variable. A manera de ejemplo, supóngase que se quiere crear una variable "estatura.cat", resultado de clasificar la estatura de las personas de acuerdo con el siguiente criterio: personas que midan menos de 150 cm serán clasificadas como "Baja"; personas entre 160 cm y 180 cm serán clasificadas como "Mediana" y personas que midan más de 180 cm serán clasificadas como "Alta". La forma de lograr esto en el lenguaje R utilizando la función `cut` se muestra a continuación:

```
# Creación de nueva variable categórica

personas$estatura.cat <-
  cut( personas$estatura, c(1.50, 1.60, 1.80, 2.10),
                          c("Baja", "Mediana", "Alta"))

personas
          estatura        peso    genero  estatura.cat
1         1.68            55       F      Mediana
2         1.84            91       M      Alta
3         1.78            88       M      Mediana
4         1.55            47       F      Baja
5         1.89            105      M      Alta
```

Debe notarse cómo en la utilización de la función `cut` el número de niveles de categorización es uno menos que el número de valores utilizados en la delimitación de cada nivel. También es posible especificar únicamente los niveles de categorización y permitir que R determine cuáles deben ser los valores de delimitación. Para clasificar la estatura,el comando utilizado para permitir que R determine los niveles de delimitación es:

```
personas$estatura.cat <-
  cut( personas$estatura, 3, c("Baja", "Mediana", "Alta"))

personas
          estatura       peso     genero        estatura.cat
1         1.68           55       F             Mediana
2         1.84           91       M             Alta
3         1.78           88       M             Alta
4         1.55           47       F             Baja
5         1.89           105      M             Alta
```

Referencias bibliográficas

[1] R Core Team, *R: A Language and Environment for Statistical Computing*. Viena: R Foundation for Statistical Computing, 2014. [En línea]. Disponible en http://www.r-project.org

[2] M. J. Crawley, "Statistics: An introduction using R", *Journal of Statistical Software*. Vol. 67, no. 5, 2nd Edition, John Wiley and Sons Inc, pp. 1-4, 2015.

[3] P. Dalgaard, *Introductory Statistics with R*, Second Edition, Springer, 2008. [En línea]. Disponible en: http://www.academia.dk/BiologiskAntropologi/Epidemiologi/PDF/Introductory_Statistics_with_R__2nd_ed.pdf

[4] S. Munzert, C. Rubba, P. Meissner y D. Nyhuis, *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining*. [En línea]. Disponible en: https://www.wiley.com/en-co/Automated+Data+Collection+with+R:+A+Practical+Guide+to+Web+Scraping+and+Text+Mining-p-9781118834817

[5] D. Nolan and D. Temple Lang, *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*, CRC Press, 2015. [En línea]. Disponible en: https://www.crcpress.com/Data-Science-in-R-A-Case-Studies-Approach-to-Computational-Reasoning-and/Nolan-Lang/p/book/9781482234817

[6] S. Stowell, *Using R for Statistics*. Apress, 2014. [En línea]. Disponible en: https://www.apress.com/gp/book/9781484201404

[7] J. M. Quick, *Statistical Analysis with R.* Beginner's Guide. Packt Publishing, 2010. [En línea]. Disponible en: https://www.amazon.com/Statistical-Analysis-John-M-Quick/dp/1849512086

[8] J. E. Friedl, *Mastering Regular Expressions*. Third Edición, O' Reilly, 2006. [En línea]. Disponible en: https://doc.lagout.org/programmation/Regular%20

Expressions/Mastering%20Regular%20Expressions_%20Understand%20Your%20Data%20and%20Be%20More%20Productive%20%283rd%20ed.%29%20%5BFriedl%202006-08-18%5D.pdf

[9] J. Ooms, D. James, S. DebRoy, H. Wickman y J. Horner, *RMySQL: Database Interface and 'MySQL' driver for R*. R package version 0.10.11, 2017. [En línea]. Disponible en: https://cran.rproject.org/package=RMySQL

[10] K. Muller, J. Ooms, D. James, S. DebRoy, H. Wickman, J. Horner, RStudio, and K. Hogskolan. *RMariaDB: Database Interface and MariaDB Driver*, R package version 1.0-2, May 6, 2018.

[11] S. Urbanek, *RJDBC: Provides access to databases through the JDBC interface*. R package version 0.2-5, 2014. [En línea]. Disponible en https://cran.r-project.org/package=RJDBC

[12] B. Ripley and M. Lapsley, *ODBC Database Access*. Package 'RODBC' version 1.3-15, 2017. [En línea]. Disponible en: https://cran.r-project.org/package=RODBC

[13] D. Mukhin, D. A. James and J. Luciani, *ROracle: OCI Based Oracle Database Interface for R*. R package version 1.3-1, 2016. [En línea]. Disponible en: https://cran.rproject.org/package=ROracle.

[14] A. Collier, feedeR: *Read RSS/Atom Feeds from R*. R package version 0.0.7, 2016. [En línea]. Disponible en: https://cran.r-project.org/package=feedeR

5. Fundamentos de aprendizaje de máquina en R

Sobre todo, no hagas daño.
—Hipócrates

El propósito de este capítulo es introducir los conceptos de aprendizaje de máquina aplicados al análisis de datos. El capítulo aborda tres temáticas: el problema de clasificación, en particular lo concerniente a los modelos de clasificación basados en el algoritmo *k-NN*; el aprendizaje probabilístico, en particular las técnicas de árboles de decisión y los modelos de segmentación o *clustering*, en particular los modelos de segmentación basados en el algoritmo *k-means*.

5.1. Introducción al aprendizaje de máquina

El aprendizaje de máquina es subcampo de la inteligencia artificial y la estadística, cuyo objetivo principal es el desarrollo de algoritmos computacionales para transformar datos en acciones inteligentes. Dentro del contexto del análisis de datos, las técnicas de aprendizaje de máquina proveen algoritmos para transformar los datos en *conocimiento accionable*, aprovechando el potencial de los datos, a través de técnicas sistemáticas que ayuden a dar sentido a los mismos y al proceso de toma de decisiones.

Esta definición está relacionada con el concepto de *minería de datos* y no es claro hasta qué punto estos dos términos se sobrelapan. En este libro utilizaremos la diferenciación, generalmente encontrada en la literatura, en la que *aprendizaje de máquina* se enfoca en lograr que los computadores aprendan cómo utilizar los datos para resolver un problema, mientras *minería de datos* se enfoca a la identificación de patrones utilizados en la solución de un problema.

Si bien es casi imposible listar todos los casos de usos y aplicaciones de las técnicas de aprendizaje de máquina en la vida cotidiana, existen numerosas y prominentes aplicaciones de estas técnicas en medicina, finanzas o climatología. Entre estas aplicaciones se pueden encontrar: predicción del resultado de elecciones populares, pronóstico del estado del tiempo y de cambios climáticos, detección de fraude en transacciones bancarias, descubrimiento de secuencias genéticas involucradas en enfermedades, e incluso, hoy día, los algoritmos que permiten que los drones y vehículos puedan conducirse sin intervención humana.

Estas aplicaciones de las técnicas de aprendizaje de máquina, dejan entrever que pueden llegar a ser más exitosas cuando son utilizadas para aumentar, en lugar de reemplazar, el conocimiento especializado que pueda llegar a tener un experto en la materia. De hecho, existe consenso en la comunidad científica referente a que el objetivo actual del aprendizaje de máquina no es la creación de un "cerebro artificial", sino utilizar estas técnicas para *asistir* a la humanidad en el entendimiento de la gran cantidad de datos con los que se cuenta, labor a todas luces imposible sin técnicas de inteligencia artificial, y en particular de aprendizaje de máquina.

Es importante entender los límites de las técnicas de aprendizaje de máquina, y en ese sentido es clave recalcar que actualmente las técnicas operan sobre parámetros estrictos, sin noción del "sentido común" y por ello es imprescindible determinar qué es exactamente lo que el algoritmo "ha aprendido" antes de aplicarlo en escenarios de la vida real. También es cierto que las técnicas vigentes todavía presentan deficiencias en algunas actividades realizadas por los humanos, como procesamiento de texto o habla e incluso a la hora de contestar preguntas sencillas. Tampoco las máquinas son, a la fecha, buenas realizando preguntas o determinando cuáles vale la pena plantear, por mencionar ejemplos de actividades simples de realizar para los seres humanos.

De hecho, fue sonado el caso de la aplicación Google Photos, que etiquetó de forma errónea a dos personas afrodescendientes y las confundió con gorilas, lo que constituyó el primer caso de racismo en inteligencia artificial. Por supuesto, el avance de la ciencia logrará que los computadores realicen estas actividades, incluso mejor que los humanos, pero por ahora vale la pena resaltar que las técnicas de aprendizaje de máquina son tan buenas como los datos de los cuales aprenden. En ese sentido el reto es ofrecer a los algoritmos buenos conjuntos de datos para el entrenamiento y la validación de los modelos.

Si bien, el principal objetivo de las técnicas de aprendizaje de máquina no es la creación de un cerebro artificial, es importante notar que el aprendizaje utilizado por estas técnicas es similar al usado por los seres humanos (y de hecho se basan en este proceso natural); este proceso puede ser dividido en cuatro componentes interrelacionados: almacenamiento de datos, abstracción, generalización y evaluación.

- *Almacenamiento de datos*: utiliza la memoria y las observaciones previas para proveer una base de hechos factuales sobre los cuales realizar razonamientos.
- *La abstracción*: involucra la traducción de los datos almacenados en conceptos, representaciones y asociaciones entre conceptos.
- *La generalización*: utiliza las abstracciones realizadas sobre los datos para crear conocimiento e inferencias que puedan ser utilizadas en contextos diferentes.

- *La evaluación*: es un mecanismo de realimentación que mide la utilidad del conocimiento aprendido e informa de potenciales mejoras.

Estos cuatro componentes están estrechamente relacionados y de hecho no se presentan de forma lineal, en el sentido de "uno después del otro". Vale resaltar que los seres humanos realizan este proceso de manera más o menos inconsciente.

Debido al enfoque utilizado en este libro, es relevante para el lector ver cómo este proceso aplica en un proyecto que involucre técnicas de aprendizaje de máquina. En general, cualquier algoritmo de aprendizaje de máquina requiere los siguientes pasos para ser desplegado:

- *Recolección de datos*. En esta parte del proceso se recolectan datos a ser utilizados por el algoritmo de aprendizaje.
- *Preparación y exploración de datos*. Proceso de exploración y análisis de calidad de los datos, así como preparación de los datos para el aprendizaje en los formatos requeridos por el algoritmo. Cualquier algoritmo de aprendizaje es tan bueno como sus datos de entrada, de forma que este proceso es fundamental para aprovechar el verdadero potencial de las técnicas.
- *Entrenamiento del modelo*. Una vez los datos estén preparados para análisis, se ejecuta el algoritmo de aprendizaje para obtener un modelo.
- *Evaluación del modelo*. Dado que el modelo aprendido suele estar un poco sesgado, es importante verificar qué tanto el método computacional aprende de su experiencia. Para ello se utiliza un conjunto de datos de prueba, para determinar el verdadero potencial del modelo en casos diferentes a los utilizados en el entrenamiento.
- *Mejoramiento del modelo*. Si se requiere mejor desempeño del modelo, deben utilizarse estrategias más avanzadas como cambiarlo, proveer más datos de entrenamiento o de pruebas o realizar un mejor trabajo a nivel de calidad de los datos.

5.2. Tipos de algoritmos de aprendizaje de máquina

Es común que los algoritmos de aprendizaje de máquina se dividan en diferentes categorías. Entender esta clasificación es quizás el primer paso a tener en cuenta si se desea convertir los datos disponibles en "conocimiento accionable", uno de los fines del aprendizaje de la máquina.

En este punto es importante retomar las definiciones dadas en el capítulo 2 del libro e introducir terminología utilizada en el aprendizaje de máquina. En primer lugar, un *modelo predictivo* busca la predicción de un valor utilizando otros valores en

el conjunto de datos. En los algoritmos de aprendizaje de máquina, este pretende *descubrir y modelar* las relaciones entre la característica objetivo y las otras *características*.

Con el fin de unificar la terminología, es fundamental aclarar que lo conocido en estadística como *variable* es entendido en el aprendizaje de máquina como *característica feature*, así como el término *observación* es comúnmente denominado como *ejemplo* en el aprendizaje. La predicción no necesariamente implica un pronóstico a futuro, si bien es la aplicación más común, ya que también pueden utilizarse en predicciones de eventos pasados o en tiempo real.

Dado que en los modelos predictivos está especificado qué se requiere aprender y cómo lo deben aprender, el proceso de entrenar un modelo predictivo se conoce como *aprendizaje supervisado*. Formalmente, *un algoritmo de aprendizaje supervisado* busca optimizar una función —el modelo— para encontrar la combinación de valores de las características que resulten en el rasgo objetivo. La tarea más común de aprendizaje supervisado es predecir en cuál *categoría* —denominada formalmente como *clase*— debe ubicarse un ejemplo, donde cada categoría tiene múltiples *niveles*. Este problema se conoce como de *clasificación*, y dada su popularidad no es de extrañar que existan múltiples algoritmos que lo aborden cada uno con sus ventajas y desventajas.

En oposición a un modelo predictivo, un *modelo descriptivo* es utilizado en tareas que se benefician de los descubrimientos obtenidos al resumir —describir— el conjunto de datos en diferentes formas. En estos modelos ninguna característica es más importante que otra y de hecho no hay una característica objetivo que tenga que ser aprendida. Debido a esto, el entrenamiento de un modelo descriptivo se denomina *aprendizaje no supervisado*. Una de sus tareas es la de *descubrimiento de patrones*, que busca identificar asociaciones útiles entre los datos, como *análisis de canasta*, cuyo objetivo es hallar ítems que frecuentemente se compran en conjunto, por ejemplo, brownies-avenas, vestidos de baño-gafas de sol o huevos-naranjas De hecho, uno de los casos originales de este de análisis es cuanto menos curioso, puesto que detectó una asociación entre ¡pañales y cervezas¡ en los consumidores de Estados Unidos.

Otra tarea común de un modelo descriptivo es la *segmentación o clustering*, consistente en dividir un conjunto de datos en grupos homogéneos llamados *clústers*. Algunas aplicaciones de esta tarea incluyen identificar individuos similares en comportamiento o información demográfica, aplicación conocida como *análisis de segmentación*. De forma similar al algoritmo de clasificación, su uso es tan común que existen diversos algoritmos que resuelven el problema.

A continuación se introducirán algunos algoritmos y su implementación en el lenguaje de programación R. Muchos de estos no se encuentran en la implementación base de la plataforma y por ello es necesario instalar paquetes adicionales como `RWeka`, el cual se instala y configura como sigue:

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

5.3. Modelos de clasificación

Estos modelos corresponden a uno de los enfoques comúnmente utilizados para resolver los "clasificadores por vecinos más cercanos" o *nearest neighbors*. La idea general, detrás de este enfoque, es sencilla y se puede resumir en el principio "si dos cosas han de ser similares, es porque sus propiedades también lo son".

Los algoritmos de clasificación por vecinos más cercanos utilizan este principio para catalogar los datos en la misma categoría en que previamente se han clasificado datos similares, justamente sus "vecinos más cercanos". Detrás de este concepto existen muchas aplicaciones, como reconocimiento de rostros, reconocimiento de texto escrito, sistemas de recomendación o incluso identificación de patrones genéticos.

Existen limitaciones a este enfoque: los algoritmos basados en "vecinos más cercanos" funcionan bien en tareas de clasificación en las cuales los ítems que pertenecen a una misma clase tienden a ser homogéneos, es decir, existen claras distinciones entre las clases o categorías. Si esta distinción no está clara, el algoritmo puede tener problemas identificando los límites de cada clase (o categoría), y por ello algunos elementos pueden quedar incorrectamente clasificados.

Los algoritmos basados en el enfoque de vecinos más cercanos se denominan *algoritmos tardíos o perezosos*, pues las fases de abstracción y generalización no se llevan a cabo de forma que el algoritmo no aprende nada. Otros autores se refieren a este tipo de algoritmos como *aprendizaje basado en instancias*, si bien el concepto es el mismo, pues no se aprende ningún parámetro sobre los datos. Por esta razón se dificulta vislumbrar cómo el clasificador está utilizando los datos. Aún así, es importante saber que pese a que un algoritmo sea tardío no implica que no sea un algoritmo poderoso y útil, como lo muestran las aplicaciones antes mencionadas, y en ese sentido merece la pena entender los detalles internos de un algoritmo basado en la técnica de vecinos más cercanos, a saber, el algoritmo *k-NN*.

5.3.1. Algoritmo kNN

Si bien es uno de los algoritmos más simples, el algoritmo *k-NN* o *k-vecinos más cercanos* es uno de los más utilizados en las tareas de clasificación, pues su fase de entrenamiento es rápida pese a que su etapa de clasificación es lenta. También tiene otras desventajas porque el algoritmo como tal no produce un modelo, luego no es tan sencillo asimilar las relaciones entre las características *features* de los objetos y las clases, relaciones que en algunas aplicaciones deben ser entendidas.

También se considera desventaja que el parámetro *k* pueda ser introducido como entrada al algoritmo sin un criterio de selección apropiado. *k* es un término variable que indica el número de vecinos más cercanos utilizado en el proceso de clasificación, y como tal desempeña papel fundamental en el algoritmo. El algoritmo *k-NN* funciona como sigue:

1. Seleccionar un valor *k*.
2. Seleccionar un conjunto de entrenamiento con suficientes ejemplos clasificados en diferentes categorías o clases en una variable nominal.
3. Luego, para cada observación del conjunto de prueba identificar los *k* ejemplos en los datos de entrenamiento que son los más cercanos en similitud, de acuerdo con una métrica de similitud establecida.
4. De estos *k* elementos más cercanos, obtener la clasificación de la mayoría y asignar esta categoría a la observación de prueba.

Según se observa, el algoritmo *k-NN* requiere una *función de distancia* que permita medir la similitud entre dos objetos. Tradicionalmente, el algoritmo utiliza la *distancia euclidiana*, no obstante otras métricas, como las distancias manhattan, canberra, minkowski, también pueden utilizarse en el entorno R.

Sean *p* y *q* dos ejemplos (objetos) que se desean comparar, cada uno de ellos con *n* características. A manera de convención, sean p_1, p_2, ⋯, p_n los valores de estas *n* características para el ejemplo *p*, y análogamente q_1, q_2, ⋯, q_n los valores de estas para el ejemplo *q*. Si las características son de tipo numérico, se define la distancia *dist*(*p*, *q*) como:

$$dist(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \qquad (5.1)$$

El valor de *k* en el algoritmo *k-NN* juega papel fundamental en el mismo, pues este determina qué tan bien el clasificador generalizará datos futuros. Si se selecciona un valor alto de *k*, el clasificador puede ignorar patrones importantes, ya que el algoritmo *tenderá* a clasificar las futuras observaciones en la clase con la mayoría de observaciones, sin importar los vecinos más cercanos. En el otro caso, un valor bajo de *k* hará que el algoritmo sea sensible a *outliers*, datos ruidosos o incluso a datos que hayan sido mal clasificados. Por ejemplo, puede suceder que al seleccionar $k = 1$ el vecino más cercano sea un ejemplo erróneamente clasificado.

En la práctica existen al menos dos enfoques para encontrar un balance y ejecutar el algoritmo con un valor de *k* apropiado. El primer enfoque consiste en hacer $k = \sqrt{N}$, donde N es el total de ejemplos de entrenamiento. Otra alternativa es entrenar el algoritmo para varios valores de *k* y seleccionar aquel con mejores resultados.

Sin embargo, el valor de *k* pierde influencia en la efectividad del algoritmo a medida que el conjunto de entrenamiento se hace más grande, por lo cual en estos conjuntos grandes el primer enfoque resulta en extremo útil.

5.3.2. Normalización

La fórmula de distancia de la ecuación 5.1 puede presentar problemas si alguna(s) de las característica(s) del ejemplo está(n) en una escala diferente a la escala de las otras características. Supóngase que una característica está medida en una escala de 0 a 10, mientras la otra está medida en una escala de 0 a 1000; las diferencias entre los valores harán que la fórmula de distancia se diferencie únicamente por la segunda característica, sin tener en cuenta las otras.

Una forma de solucionar este problema es mediante un reescalamiento de cada valor de la característica. El método tradicional de reescalamiento es la *normalización min-max*. Este es un proceso de transformación que hace que todos los datos se encuentren en el intervalo [0, 1], y puede interpretarse como un porcentaje (de 0% a 100%) de qué tan lejos el valor original se encuentra en el rango de la variable. La fórmula de transformación para un conjunto de datos *X* es la siguiente:

$$X_{nuevo} = \frac{X - min(X)}{max(X) - min(X)} \tag{5.2}$$

Una forma de implementar esto en R es definiendo una función de normalizacion `normalizacion.min.max`, la cual sigue directamente la enunciación antes dada.

```
normalizacion.min.max <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}
x <- rnorm(50)

normalizacion.min.max(x)
[1] 0.47866755 0.55107688 0.68179903 0.41813673 0.63475738 0.32035425
[7] 0.58386657 0.07191344 0.40919836 0.48270735 0.49920988 0.44249158
[13] 0.34936539 0.61718382 0.65149824 0.56885081 0.55358324 0.46788009
[19] 0.98457248 0.54164045 0.66068095 0.79458481 0.54695811 0.65937930
[25] 0.42112569 0.16778615 0.55245682 0.63925279 0.31927690 0.22865491
[31] 0.83696501 0.51477526 1.00000000 0.00000000 0.69775077 0.29212509
[37] 0.36627155 0.14088320 0.71351510 0.95141488 0.60551371 0.45180782
[43] 0.40734005 0.44167220 0.73641486 0.45739304 0.23351581 0.24288683
[49] 0.66123040 0.44047764
```

Otra medida de transformación es la *estandarización por unidad tipificada*, más conocida por su nombre en inglés *z-score*. Esta es una medida de normalización común en análisis estadístico y se define como sigue:

$$X_{nuevo} = \frac{X - \mu}{\sigma} \tag{5.3}$$

De acuerdo con la notación seguida en el libro, debe recordarse que μ representa la media y σ la desviación estándar de la variable. A diferencia de la normalización max-min, esta tiene valores tanto positivos como negativos y no tienen un valor mínimo o máximo.

En la plataforma R esta transformación ya está definida como parte de la instalación básica. La funcionalidad la realiza la función `scale` y se utiliza como sigue:

```
wx <- rnorm(50)
> scale(x)
  [,1]
[1,] -1.60158521
[2,] 0.44456316
[3,] 0.23765034
...
[48,] -0.47182806
[49,] -0.34309173
[50,] 0.04694912
attr(,"scaled:center")
[1] 0.2368786
attr(,"scaled:scale")
[1] 0.8366277
```

Un comentario final sobre normalización, fundamental en el proceso de clasificación, tiene que ver con que los valores de los nuevos ejemplos pueden estar por fuera de los valores utilizados en el entrenamiento, por lo cual el proceso de normalización de los datos de pruebas o de los nuevos valores puede estar por fuera del rango original. Debido a ello, se recomienda definir dos valores máximos y mínimos constantes —garantizando que todos los valores se encuentren en este rango— y utilizar estos valores en las fórmulas de normalización.

También es importante aclarar que si se utiliza la estandarización por unidad tipificada, los futuros ejemplos deben tener una media y una desviación estándar similar a los ejemplos utilizados en el entrenamiento. Esta validación puede realizarse utilizando las pruebas de hipótesis explicadas en la sección 2.3.

5.3.3. Algoritmo k-NN en la plataforma R

Al ser un algoritmo comúnmente utilizado, no es de extrañar que ya esté implementado en la plataforma R. Una de las varias aplicaciones del algoritmo se encuentra en

el paquete `class`, por lo que el primer paso debe ser instalar este paquete y cargarlo en el entorno R.

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

La aplicación del paquete `class` es una implementación estándar del algoritmo utilizando distancias euclidianas. Los procesos de entrenamiento y clasificación emplean una sola función `knn` de la siguiente forma:

```
knn( train, test, cl, k)
```

Los parámetros requeridos por la función `knn` son:

- El primer parámetro de la función es un data frame que contiene el conjunto de datos numéricos a utilizarse en el entrenamiento.
- El segundo parámetro es un data frame que contiene el conjunto de datos numéricos a utilizar en la etapa de pruebas.
- El tercer parámetro es un *vector* con la categoría de cada ejemplo en los datos de entrenamiento.
- El cuarto parámetro es el entero *k*, que indica el número de vecinos más cercanos a determinar por el algoritmo.

La función retorna un *vector* con la predicción de la categoría para cada ejemplo de los datos de prueba. Supóngase que se cuenta con los conjuntos de datos `datos_entrenamiento`, `datos_prueba` y con el vector de categorías `categorías_entrenamiento`. Según la explicación anterior, una forma de utilizar la función `knn` es:

```
pred <- knn( train = datos_entrenamiento,
test = datos_prueba,
cl = categorias_entrenamiento,
k = 8)
```

Naturalmente, el valor del parámetro *k* se ingresa de acuerdo con las técnicas anteriormente expuestas, teniendo en cuenta la influencia de esta variable en la eficacia general del algoritmo.

En la práctica, es común utilizar 75 % de los ejemplos para el entrenamiento y 25 % para la validación, de forma que si se tiene el conjunto de datos, digamos en una variable `datos`, los subconjuntos se obtienen como sigue:

```
tam <- nrow(datos)
datos_entrenamiento <- datos[1:tam*0.75]
datos_prueba <- datos[tam*0.75+1:tam]
```

Una vez se cuenta con la predicción de la categoría para cada ejemplo de prueba, el siguiente paso es validar el modelo. En este punto se conocen tanto los resultados de las categorías originales del conjunto de pruebas, como las categorías resultados de la predicción, luego el proceso consiste en evaluar qué tanto concuerdan ambos vectores de valores. Esta comparación se puede realizar de múltiples formas, siendo la más frecuente el uso de *tabulaciones cruzadas*. R provee, aparte de las funciones de la clase `table` explicadas en el capítulo 3, el paquete adicional `gmodels`, que incluye una función de tabulación cruzada más apropiada para esta parte del proceso.

```
install.packages("gmodels")
```

La función a utilizar del paquete `gmodels` se denomina `CrossTable` y recibe por parámetro tres argumentos: un vector con las categorías originales, otro con las categorías de la predicción y un tercero estadístico relacionado con algunos ajustes de la distribución "chi-cuadrado".

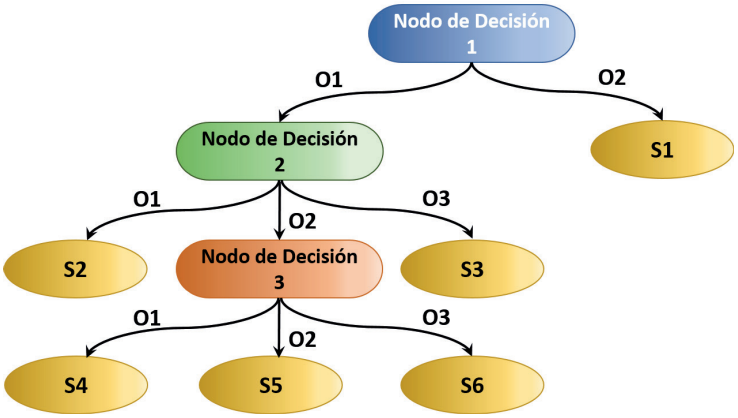
```
CrossTable( x=cat_original, y=pred,prop.chisq=FALSE)
```

La tabulación cruzada ofrece por resultado la *precisión del modelo*, obteniendo un porcentaje del total de categorías correctamente clasificadas, esto es, que concuerdan. Entre más cercano sea este valor al 100% se considera que el modelo, al menos en los datos de entrenamiento, es mejor.

5.4. Árboles de decisión y de clasificación

La clasificación por árboles de decisión utiliza una *estructura de árbol* para modelar las relaciones entre las características y las potenciales salidas del modelo. La estructura del árbol tiene los siguientes componentes: *nodos de decisión*, que requieren decidir sobre múltiples *opciones u alternativas* basadas en un atributo; cada una de estas alternativas divide los datos en *ramas* que indican las potenciales *salidas* de una decisión. Las alternativas pueden ser binarias —sí o no— o comprender más de dos posibilidades. En caso que una decisión final pueda determinarse, la rama termina en un *nodo hoja o terminal*, que denota la acción a realizar como resultado de elecciones efectuadas desde la *raíz* del árbol hasta el nodo terminal. Un ejemplo de la estructura de un árbol de decisión se muestra en la figura 5.1.

Figura 5.1. Ejemplo de árbol de decisión



Fuente: Elaboración propia

Una de las ventajas como clasificador de un árbol de decisión respecto a un algoritmo *k-NN* es que en un árbol de decisión es claro cómo y por qué el algoritmo funciona para una tarea particular, mientras que en el algoritmo *k-NN* no es del todo transparente el mecanismo de clasificación. Esto hace que estos clasificadores resulten útiles en diagnóstico de enfermedades, procesos de aplicación de candidatos a créditos o admisión en alguna institución académica, en estudios de mercadeo, y en general cualquier estudio que demande compartir resultados con otros de forma que la construcción del modelo de clasificación requiera estar explícita.

Es importante mencionar que a pesar de su utilidad, existen escenarios en los cuales no es del todo conveniente utilizar árboles para las tareas de clasificación. En particular, aquellas aplicaciones con gran número de características nominales con múltiples niveles o con gran cantidad de características numéricas. La principal razón detrás de esto es la complejidad asociada a gran número de nodos de decisión y a cierta tendencia a sobreajustar los datos, reduciendo de esta forma la capacidad predictiva del modelo.

5.4.1. Construyendo un árbol de decisión: algoritmo C5.0

Dado que los datos del mundo real contienen múltiples características por elemento, los árboles de decisión rápidamente alcanzan un nivel de complejidad alto, con numerosos nodos de decisión, ramas y nodos terminales. Debido a esto, se planteó la posibilidad de *construir* de *forma automática* los modelos basados en árboles de decisión. En este libro se utilizará el algoritmo `C5.0`, desarrollado por J. Ross Quinlan; versión mejorada de su algoritmo `C4.5` y que se considera el estándar “” de facto para construir árboles de decisión. Las razones detrás del porqué este algoritmo se ha

convertido en el estándar "de facto" son variadas, pero principalmente tienen que ver con que funciona bien para un número considerable de problemas sin requerir mayor configuración, además de lo sencillo que es de entender, ya que puede ser interpretado sin requerir demasiada formalidad matemática. Otras características importantes de este algoritmo son:

- Maneja correctamente características numéricas y nominales, así como datos faltantes.
- Puede ser utilizado en conjuntos de datos grandes y pequeños.
- Excluye características que no son importantes.
- Comparado con otros modelos como redes neuronales o máquinas vectoriales es más eficiente.

Sin embargo, el algoritmo presenta debilidades, las cuales deben ser tenidas en cuenta si se ha de utilizar en algún proyecto práctico. Las principales limitaciones del algoritmo son:

- Tendencia a obtener un modelo que sobre-ajusta o sub-ajusta los datos.
- Pequeños cambios en los ejemplos de entrenamiento pueden cambiar sustancialmente la lógica de decisión.
- En árboles de decisión de gran complejidad las decisiones pueden parecer contrarias a la intuición y difíciles de interpretar.
- Tendencia a que las características nominales con mayor número de niveles dominen la lógica de decisión en el modelo.

Con las ventajas y debilidades del algoritmo C5.0 identificadas, se detallan a continuación los pasos involucrados en la construcción automática de un árbol de decisión. Sin embargo, como paso preliminar a la explicación, es necesario introducir terminología y definiciones matemáticas en las cuales está basado el algoritmo.

Pureza es una medida de qué tanto un subconjunto de ejemplos contiene únicamente una clase. Cualquier subconjunto compuesto solo de una clase se denomina *puro*. Estas definiciones son importantes porque uno de los objetivos del algoritmo es particionar el conjunto de datos, de forma que se obtengan subconjuntos *puros*, cuyos ejemplos pertenezcan a una única clase.

Si bien existen varias medidas de *pureza*, el algoritmo C5 hace uso del concepto de *entropía*. La *entropía* es una medida que cuantifica la aleatoriedad, desorden u homogeneidad de un conjunto de valores en una clase. Los conjuntos con alta entropía son ampliamente diversos, en tanto un elemento del conjunto ofrece poca información sobre los elementos restantes del mismo. En contraparte, los conjuntos

con baja entropía son ampliamente homogéneos. La unidad de medida de la entropía se denomina *bits*.

Sea *S* un (sub)conjunto de datos, *c* el número de niveles de una característica de tipo categórico (o número de clases) y p_i la proporción de valores del (sub)conjunto de datos clasificados en el nivel *i*. Se define la *entropía* como:

$$Entropia(S) = \sum_{i=1}^{n} -p_i log_2(p_i) \tag{5.4}$$

Para *n* clases, la entropía se encuentra entre los valores 0 a $log_2 n$ bits. Una entropía de 0 bits significa que el conjunto es totalmente homogéneo (puro), mientras el valor máximo de $log_2 n$ indica que el conjunto es tan diverso como es posible.

Un reto fundamental a resolver en la construcción automática de un árbol de decisión a partir de un conjunto de datos, es identificar la característica involucrada en el nodo de decisión para generar las distintas alternativas. Para determinar esta característica, el algoritmo utiliza la métrica de entropía, previamente definida, para calcular el *cambio en la homogeneidad* del (sub)conjunto de datos que resultaría de particionar el (sub)conjunto utilizando *cada una* de las características posibles. Esta medida se denomina *ganancia de información* o *Information Gain* (IG), y se define como:

$$IG(C) = Entropia(S_1) - Entropia(S_2) \tag{5.5}$$

donde *IG(C)* es la ganancia de información para la característica *C*, S_1 es el segmento antes de la partición, y S_2 las particiones resultantes después de la partición del (sub) conjunto. Sin embargo, después del particionamiento los datos son divididos en más de una partición, luego *Entropia*(S_2) debe considerar la entropía total teniendo en cuenta todas estas particiones. Para ello, defínase w_i como la proporción de ejemplos en cada partición P_i y defínase la entropía total de todas las particiones como una suma ponderada, así:

$$Entropia(S_2) = \sum_{i=1}^{n} w_i Entropia(P_i) \tag{5.6}$$

A mayor ganancia de información, se puede concluir que esa característica es la mejor, creando grupos homogéneos después de una partición al utilizar dicha característica. Si no hay ganancia de información, se puede concluir que no habría reducción en la entropía del (sub)conjunto si se particionara el conjunto de datos por esta característica, luego no sería una característica candidata a utilizar en el particionamiento. Por otro lado, si la entropía, después de la partición del conjunto (*Entropia*(S_2)) es igual a 0, significa que la partición obtiene por resultado grupos completamente homogéneos.

5.4.2. Árboles de decisión en la plataforma R

La función a utilizar en R para crear árboles de decisión utilizando el algoritmo C5.0 es la `c5.0`, que recibe los siguientes parámetros:

- Tabla de datos con el conjunto de entrenamiento.
- Para cada ejemplo (observación) del conjunto de entrenamiento, el vector con la clase a la cual pertenece cada uno de ellos.
- Un número opcional que controla el número de iteraciones del proceso.
- Una matriz opcional con información de manejo de error.

La función se utiliza en dos etapas: la primera crea el modelo haciendo uso de la función C5.0.

```
modelo < - C5.0( datos, categorias )
```

La segunda, similar a los métodos de clasificación, utiliza la función `predict` para validar el modelo sobre los datos de entrenamiento.

```
pred < - predict( modelo, datos_pruebas )
```

La función `summary` sobre el modelo contiene toda la información del modelo, incluida la estructura del árbol de decisión. De forma similar a los algoritmos de clasificación se pueden utilizar tabulaciones cruzadas para determinar la precisión del modelo.

5.5. Modelos de segmentación

Son una técnica cuyo objetivo es encontrar de forma automática subgrupos de observaciones en un conjunto de datos. Si bien pueden ser utilizados como una técnica de reducción de un número grande de observaciones a un número menor de grupos o "clúster", su principal aplicación tiene que ver con hallar grupos de observaciones que bajo cierta métrica se puedan considerar similares. Esto tiene aplicaciones en campos como el mercadeo, donde se pueden agrupar los compradores en grupos similares basados en sus hábitos de compra o en su información demográfica. De esta forma, es posible diferenciar y enfocar de forma estratégica las campañas de mercadeo para cada uno de estos subgrupos.

A cada subgrupo se denomina *clúster*, término que informalmente se define como un grupo de observaciones más similares entre ellas de lo que son a las observaciones realizadas en otros grupos. Dependiendo de la forma en la cual se arman los *clúster*, las técnicas se pueden dividir en técnicas de *particionamiento*, en las que se especifica a priori el número de *clúster K* y estos "migran" observaciones entre

uno y otro formando grupos uniformes o en técnicas *jerárquicas* aglomerativas, en las que cada observación comienza siendo su propio *clúster* y después estos empiezan a combinarse. Para cada técnica, existen diversos algoritmos, siendo los más populares "*k-means*", en el caso de las técnicas de particionamiento, y los algoritmos *vinculados* (enlazados) en el caso de las técnicas jerárquicas.

5.5.1. Proceso de creación de modelos de segmentación

Antes de invocar las respectivas funciones en la plataforma de análisis de datos es necesario seguir una serie de pasos comunes para que la construcción del modelo de segmentación sea efectiva.

1. Seleccionar los atributos apropiados que permitan identificar diferencias entre las distintas observaciones del conjunto de datos. Este es quizás el paso más importante en el análisis, pues una selección inapropiada de las variables hará que los algoritmos no arrojen los resultados esperados del análisis.
2. Normalizar los datos siguiendo los procesos de transformación explicados en la sección 5.3.2. Tanto la normalización min-max como el *z-score* son frecuentemente utilizados en los análisis de segmentación.
3. Identificar los *outliers*, ya que muchos algoritmos son en extremo sensibles a los datos aislados. También pueden utilizarse algoritmos más robustos a la presencia de estos datos.
4. Definir la métrica de distancia entre las entidades a segmentar.
5. Seleccionar el algoritmo de segmentación. Generalmente se escogen varios métodos para identificar qué tan robusto es el resultado dependiendo de la elección del algoritmo.
6. Obtener varias soluciones.
7. Determinar el número de grupos o *clúster*.
8. Visualizar los resultados.
9. Interpretar los resultados.
10. Validar los resultados para evaluar no solo su estabilidad, sino determinar si tienen sentido en el campo particular de aplicación de los datos.

5.5.2. Algoritmo k-means

Es uno de los métodos más populares para el análisis de segmentación mediante técnicas de particionamiento (junto con el algoritmo PAM). Los conceptos principales alrededor de los cuales funciona el algoritmo son los de centroide y la métrica de

distancia o similitud. Informalmente se define un centroide como el vector de valores alrededor de los cuales se encuentran los elementos pertenecientes a cada subgrupo.

El algoritmo sigue estos pasos:

- Seleccionar aleatoriamente *K* observaciones del conjunto de datos cuyo objetivo de servir de *centroides*.
- Para cada observación determinar cuál es su centroide más cercano.
- Recalcular los centroides de cada *clúster*. Para ello tomar todas las observaciones pertenecientes al *clúster*, y para cada variable de la observación calcular la media. Con esto se obtiene un vector de longitud igual al número de variables, con cada componente en la correspondiente media de las observaciones del subgrupo.
- Reasignar las observaciones a su centroide más cercano.
- Repetir los pasos 3 y 4 hasta que el algoritmo alcance un número de iteraciones determinado.

En la implementación, algunos detalles pueden variar.

El algoritmo *k-means* tiene características que lo hacen relevante, como permitir conjuntos de observaciones más grandes que los encontrados en lo enfoques jerárquicos. Que las observaciones no permanezcan en el mismo *clúster* durante la ejecución del algoritmo ayuda a obtener una mejor solución. Sin embargo, tiene restricciones y desventajas, como circunscribir su aplicación a variables numéricas y que sean en extremo sensibles a valores aislados (*outliers*), consecuencia de utilizar la media aritmética en el recálculo de los centroides.

5.5.3. Modelos de segmentación en R

La función en R para realizar modelos de segmentación basados en el algoritmo `k-means` es `kmeans(df,k)`, donde `df` es el conjunto de datos numéricos y *k* es el número de subgrupos o clústers a obtener. Esta función se encuentra en el paquete `stats`.

```
grupos <- kmeans( df, k )
```

La función retorna un objeto de tipo `cluster` con la siguiente información:

- `grupos$clusters`: vector de las asignaciones a los clústers realizados por la función.
- `grupos$centers`: matriz que indica los valores medios para cada una de las características
- `grupos$size`: lista el número de ejemplos asignados a cada clúster.

A diferencia de los modelos de clasificación, en el problema de segmentación no se obtiene una predicción, en el sentido estricto de la palabra, pues no se utilizó un conjunto de entrenamiento debido a su característica no supervisada.

Referencias bibliográficas

[1] B. Lantz, *Machine Learning with R*. 2nd Edition. UK. Birmingham: Packt Publishing, 2013. [En línea]. Disponible en: http://bit.ly/2JyjycM

[2] R. I. Kabacoff, *R in Action*. Second Edition, Manning Publications Co., 2015. [En línea]. Disponible en: http://bit.ly/2LVVkuO

[3] E. Mayor, *Learning Predictive Analytics with R*. Packt Publishing, 2015. [En línea]. Disponible en: https://amzn.to/2JBlzVP

[4] J. Ledolter, *Data Mining and Business Analytics with R*. New Jersey: John Wiley Sons, 2013. [En línea]. Disponible en: http://bit.ly/2XQyszd

[5] N. Zumel and J. Mount, *Practical Data Science with R.* Manning Publications Co., 2014. [En línea]. Disponible en: http://bit.ly/2XO0M5e

6. Caso de estudio: reglas de asociación

Lo que se oye se olvida, lo que se ve, se recuerda y lo que se hace se aprende.
—Proverbio chino

En los capítulos anteriores se presentaron diferentes temáticas relacionadas con *ciencia de datos*, abarcando desde la definición del término hasta el análisis estadístico y de aprendizaje de máquina, pasando por las etapas que constituyen un proyecto de ciencia de datos. Este capítulo presenta la aplicación de algunas técnicas estudiadas, las etapas de un proyecto de ciencia de datos y nuevas técnicas de análisis de datos para resolver problemas de *reglas de asociación*, cuyo ejemplo más común es el *análisis de canasta*.

El descubrimiento de reglas de asociación se identificó como una de las aplicaciones más comunes y útiles en pequeñas y medianas empresas como parte del proyecto de investigación: *Modelos para el aporte eficaz de la ciencia de datos a la mejora de la competitividad del sector empresarial colombiano*, proyecto en el cual se enmarca este libro. Ahora, si bien los conjuntos de datos utilizados en el mencionado proyecto no son exactamente los mostrados en las secciones de este capítulo, las técnicas de análisis son esencialmente las mismas, aplicadas sobre un conjunto de datos más genérico y propicio para un libro como este.

6.1. Reglas de asociación

Estas reglas permiten explorar las relaciones entre ítems y conjuntos de ítems, como las palabras contenidas en oraciones, los componentes de productos alimenticios, o el caso más conocido, transacciones en una tienda, aplicación denominada *análisis de canasta*. Este tipo de análisis permite investigar si dos o más productos están siendo comprados juntos en la misma transacción, así como identificar si la compra de uno o varios productos incrementa la posibilidad de comprar otro.

Considérense las siguientes transacciones —compras de productos— por parte de varios usuarios:

- T_1. Huevos; jugo de naranja; bebida gaseosa; pan de molde.
- T_2. Huevos; harina de trigo; azúcar.
- T_3. Huevos; jugo de naranja; pan de molde.

- T_4. Jugo de naranja; pan de molde; azúcar.
- T_5. Huevos; jamón; pan de molde.

A primera vista parece existir alguna relación entre compras de "Huevos" y "Pan de molde", ya que 60% de las transacciones contienen ambos elementos; si bien cabe preguntarse si sería posible establecer estas relaciones en miles de transacciones, cada una de ellas compuesta por decenas de elementos. El objetivo es, entonces, obtener reglas de asociación en forma automática —incluidas aquellas que pueden no ser obvias en un primer análisis del conjunto de transacciones— así como derivar indicadores de la confiabilidad de esas asociaciones.

Una definición formal de una *regla de asociación* $X \Rightarrow Y$ establece dos componentes: un *antecedente* X, el cual es un conjunto *itemset*, compuesto por uno o varios elementos y un *consecuente* Y, el cual es un solo elemento. Un conjunto de elementos es *frecuente* si su ocurrencia es superior a un umbral denominado *soporte o apoyo mínimo*. Aquellos elementos cuya ocurrencia no supera el umbral son omitidos, proceso que se denomina *poda*.

El *soporte* de un conjunto de elementos se define como la proporción sobre el total de transacciones o casos en los cuales los elementos del conjunto están presentes. En los ejemplos anteriores, el soporte del conjunto *Jugodenaranja, pandemolde* es 0,6, pues este conjunto de elementos aparece en tres de las cinco transacciones. De igual modo, el soporte del conjunto *Huevos* es 0,8, pues está presente en cuatro de las cinco transacciones.

Otra medida de interés encontrada en la literatura de reglas de asociación es la *confianza*, definida como la proporción de los casos de X *donde* $X \Rightarrow Y$. Esta medida puede ser computada como el número de casos (transacciones) que tienen tanto a x como a Y dividido entre el número de casos en los cuales está presente únicamente X, esto es confianza ($X \Rightarrow Y$) = soporte ($X \cup Y$) / soporte (X). A manera de ejemplo, considérese la regla de asociación *Jugodenaranja, pandemolde* ⇒ *Huevos*. La confianza de esta regla es 2/3 = 0,66.

Para finalizar lo relacionado con la terminología se define el *incremento de confianza* (*lift*) como sigue: *lift*($X \Rightarrow Y$) = *soporte*($X \Rightarrow Y$) / (*soporte*(X) * *soporte*(Y)). Esta medida es indicador de qué tanto se incrementa el soporte de una regla por encima de lo que puede ser esperado únicamente por el azar; si este valor es mayor que 1, se concluye que la regla de asociación explica la relación entre los ítems mejor de lo esperado únicamente por el azar.

6.2. Algoritmo a priori

Esta sección establece los fundamentos teórico-prácticos del algoritmo `apriori`, utilizado para computar tanto los conjuntos de ítems como las reglas de asociación

de una forma computacionalmente eficiente, así como las medidas de *soporte y confianza* de estas reglas. La estrategia utilizada por el algoritmo consiste en descartar conjuntos de ítems infrecuentes sin computar su soporte. Adicionalmente, la estrategia aprovecha que los subconjuntos de un conjunto de ítems frecuente también son frecuentes.

Considérese una representación binaria de la base de datos de transacciones utilizada como ejemplo en la sección anterior. Esta representación identifica con 1 si el ítem aparece en la transacción y con 0 en caso contrario.

Tabla 6.1. Representación binaria de transacciones

Transacción	Huevos	Jugo de Naranja	Bebida Gaseosa	Pan de Molde	Harina de Trigo	Azúcar	Jamón
1	1	1	1	1	0	0	0
2	1	0	0	0	1	1	0
3	1	1	0	1	0	0	0
4	0	1	0	1	0	1	0
5	1	0	0	1	0	0	1

Fuente: Elaboración propia

El algoritmo `apriori` genera las reglas computando las posibles reglas de asociación que tienen a un solo ítem como consecuente. Para cada regla calcula la confianza, y aprovechando que si dos reglas $X1 \Rightarrow Y1$ y $X2 \Rightarrow Y2$ tienen una confianza alta, la regla $X1 \bigcup X2 \Rightarrow \{Y1, Y2\}$ también la tendrá. De esta forma, el algoritmo construye reglas con 2, 3 y demás ítems como consecuente.

6.3. Caso de estudio: reglas de asociación en R

Una vez finalizado el marco conceptual básico sobre reglas de asociación, el objetivo de esta sección es ejemplificar la aplicación de estos conceptos en un caso práctico. La minería de reglas de asociación fue una de las principales técnicas utilizadas en este proyecto de investigación, enfatizando la aplicación de estas técnicas en el diseño de los *sistemas de recomendación*.

Los *sistemas de recomendación* son herramientas de software que proveen sugerencias de ítems (información o productos), probablemente de utilidad para un usuario. Aplicaciones actuales de los sistemas de recomendación pueden encontrarse en el comercio electrónico, donde se recomiendan nuevos ítems de compra con base en lo que otros compraron o ítems que por su propio contenido se pueden considerar similares.

Otras aplicaciones se pueden encontrar en el entretenimiento como recomendaciones de videos, noticias, canciones u otro material multimedia. También se encuentran aplicaciones en la gestión del talento humano, donde se recomienda a los individuos, empresas o trabajos a los cuales aplicar, y a las empresas, individuos a contratar, o incluso, en redes sociales, sugerir amigos o personas o empresas a seguir.

Si bien en el mencionado proyecto se usaron conjuntos de datos de validación diferentes a los utilizados a continuación, las técnicas e ideas expuestas son en esencia las mismas. Adicionalmente, se involucran diferentes técnicas y análisis en la plataforma R tratadas en capítulos anteriores, con el objetivo de abordar las diferentes etapas de un proyecto de análisis de datos.

La librería R, que se utilizará en este capítulo, se denomina `arules`, la cual, aparte de proveer una implementación del algoritmo a *priori*, provee un conjunto de datos denominado `Groceries`, con el objetivo de validar el algoritmo en la aplicación de *análisis de canasta*.

Los siguientes comandos R instalan la librería y cargan el conjunto de transacciones `Groceries`.

```
# Instalar arules y cargar el data frame
# Groceries
install.packages("arules")
library(arules)
data(Groceries)
```

6.3.1. Análisis exploratorio del caso de estudio

Debido a que los procesos de obtención y limpieza de datos de la metodología (OSEMN) ya fueron realizados, se empezará el análisis exploratorio de este conjunto de datos. Algunos pasos iniciales de esta exploración involucran determinar el tamaño del conjunto de datos y los nombres de las columnas. Los comandos en R para realizar este proceso se muestran a continuación:

```
# Dimensiones del conjunto de datos
> Groceries
transactions in sparse format with
9835 transactions (rows) and
169 items (columns)
> dim(Groceries)
[1]    9835    169
> colnames(Groceries)
  [1]   "frankfurter"            "sausage"
  [3]   "liver loaf"             "ham"
  [5]   "meat"                   "finished products"
  [7]   "organic sausage"        "chicken"
  ...
```

Según lo observado, se cuenta con un total de 9835 transacciones y 169 diferentes productos en la canasta. Como es de esperarse por la aplicación, los nombres de los productos (columnas) corresponden a elementos típicos que se encontrarían en un supermercado o tienda.

Conforme a lo planteado anteriormente es común establecer estadísticas descriptivas del conjunto de datos. En este caso, estas ayudan a responder preguntas como: ¿cuál es el máximo de productos o ítems que se adquirieron en una transacción?, ¿cuántos productos en promedio se compran por transacción?, ¿cuál es el máximo de productos que llevan el 90% de quienes compran? Con ayuda de las funciones R, `summary` y `quantile`, resolver estas preguntas es sencillo.

```
# Obtener estadísticas de resumen
sizes <- size(Groceries)
summary(sizes)
  Min.    1st Qu.   Median        Mean    3rd Qu.         Max.
  1.000   2.000     3.000         4.409   6.000           32.000
# Calcular cuantiles
quantile(sizes, probs=seq(0,1,0.1))
   0%   10%    20%    30%    40%    50%    60%    70%    80%    90%    100%
    1    1       1    2      3      3      4      5      7      9      32
quantile(sizes, probs=c(0.99,1))
  99%     100%
  16      32
```

Las salidas arrojadas por R indican que en promedio cada transacción consta de tres ítems, y que el máximo de ítems en una transacción es 32. Más interesante aún, el análisis exploratorio de datos permite concluir hechos como que el 90% de las transacciones tiene menos de nueve ítems, que el 99% de las transacciones implica menos de 16 ítems o que el 22% de las transacciones contiene un único ítem. Puede hacerse uso de la visualización (en este caso, una gráfica de barras) para resaltar estos hechos.

```
# Calcular cuantiles y visualizar
# en gráfica de barras
    barplot( quantile(sizes, probs=seq(0,1,0.01)) )
```

El análisis exploratorio de datos puede incluso proveer información sobre la frecuencia relativa de cada producto (que no es otra cosa que el *soporte* del ítem individual). Preguntarse cuáles son los diez productos que más veces aparecen en las transacciones puede resolverse haciendo uso de funciones R como `itemFrequency` y `sort`.

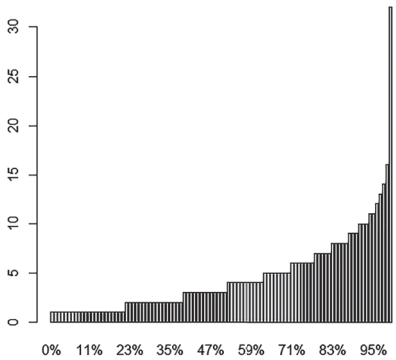
También es posible apoyarse en la visualización para determinar, por ejemplo, la distribución de los productos.

```
# Obtener la distribución de frecuencias
freqs <- itemFrequency(Groceries)
summary(freqs)
Min.       1st Qu.     Median      Mean        3rd Qu.      Max.
0.0001017   0.0038637   0.0104728   0.0260915   0.0310117    0.2555160

# Obtener el conteo de los productos
conteo <- (freqs/sum(freqs))*sum(sizes)
summary(conteo)
Min.      1st Qu      Median     Mean          3rd Qu.         Max.
1.0         38.0       103.0      256.6          305.0         2513.00

# Ordenar y obtener los 10 con más apariciones
prod.orden <- sort(conteo, decreasing=T)
prod.orden[1:10]
whole milk      other vegetables         rolls/buns           soda
   2513             1903                   1809               1715
yogurt     bottled water       root vegetables         tropical fruit
 1372         1087                  1072                    1032
shopping bags                        sausage
     969                              924
```

Figura 6.1. Distribución de cuantiles en el caso de estudio

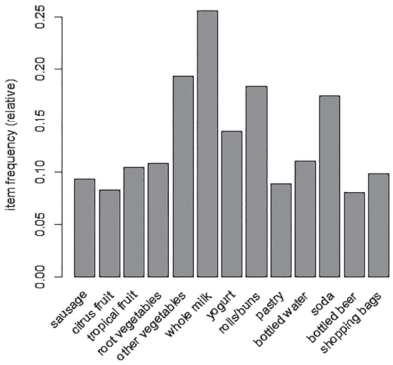


Fuente: Elaboración propia

Al observar las salidas de las funciones R se colige que la respuesta a la pregunta "¿cuáles son los diez productos que más veces aparecen en las transacciones?", son: leche entera (2513 apariciones), otros vegetales (1903), ..., yogurt (1372), agua embotellada (1087), ..., y salchichas (924). Adicionalmente, es posible generar una visualización de la frecuencia relativa de los productos con más apariciones. Para ello se utiliza la función `itemFrequencyPlot`, la cual suele recibir por parámetro el nivel mínimo de soporte.

```
# Graficar los productos con determinado
# soporte
itemFrequencyPlot(Groceries, support=0.08)
```

Figura 6.2. Frecuencia relativa de los ítems del caso de estudio



Fuente: Elaboración propia

6.3.2. Generación de modelos del caso de estudio

El análisis exploratorio de los datos provee información y visualizaciones que permiten obtener información del conjunto de datos no disponible en un primer momento, y siguiendo con la etapa de modelado de un proyecto de análisis de datos, se busca obtener algún modelo que permita entender las relaciones entre los datos y determinar predicciones. En este caso, la etapa de modelado da por resultado reglas de asociación utilizando el algoritmo `apriori`.

Como es de esperarse, la librería `arules` contiene ya una implementación del algoritmo en la función `apriori`, que recibe por parámetro el conjunto de transacciones y parámetros propios de algoritmos como el *soporte y la confianza*.

```
# Generar reglas con algoritmo apriori
# Soporte mínimo = 0.05
# Confianza = 0.1
reglas <- apriori(Groceries, parameter=
  list(support = 0.05, confidence= 0.1))
Apriori
Parameter specification:
confidence     minval    smax       arem     aval original       Support
    0.1         0.1         1         none         FALSE           TRUE
maxtime     support     minlen       maxlen        target           ext
    5         0.05         1            10         rules          FALSE
Algorithmic control:
filter    tree    heap      memopt       load       sort         verbose
0.1     TRUE     TRUE     FALSE         TRUE          2          TRUE
Absolute minimum support count: 491
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)]
done [0.00s].
sorting and recoding items ... [28 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [14 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
```

La salida de la función `apriori` provee información variada sobre algunos parámetros de control del algoritmo, el tiempo que tomó la creación de las reglas y cuántas reglas de asociación fueron creadas. En este caso, para un nivel de soporte de 0;05 y una confianza de 0;1 se crearon 14 reglas.

Para obtener las reglas de asociación generadas por el algoritmo "apriori" se utiliza la función `inspect`.

```
# Inspeccionar reglas de asociación
inspect(reglas)
      lhs                 rhs              support           confidence
...
[9]   {yogurt}       => {whole milk}       0.05602440       0.4016035
[10]  {whole milk}   =>  {yogurt}          0.05602440       0.2192598
[11]  {rolls/buns}   => {whole milk}       0.05663447       0.3079049
[12]  {whole milk}   => {rolls/buns}       0.05663447       0.2216474
...
[14]  {whole milk}   => {other vegetables}0.07483477    0.2928770
```

Haciendo uso de la función `sort`, es posible listar solo las *n* reglas de asociación que tengan el nivel más alto de confianza. Para lograr esto, se invoca la función `inspect`, como se muestra a continuación:

```
# Reglas ordenadas por confianza
inspect(sort(reglas, by="confidence"))
lhs                  rhs                  support              confidence
{yogurt}          => {whole milk}         0.05602440            0.4016035
{other vegetables} => {whole milk}        0.07483477            0.3867578
{rolls/buns}      => {whole milk}         0.05663447            0.3079049
{whole milk}      => {other vegetables}   0.07483477            0.2928770
```

En este caso, la regla con más confianza indica que "Yogurt" y "Leche entera" tienden a estar juntos en las transacciones. Sin embargo, queda por resolver cómo se realiza la validación del modelo, una de las últimas etapas de la metodología (OSEMN).

Para validar el modelo se hace uso de la métrica "*incremento de confianza*" o *lift*. Como se introdujo en la sección 6.1., un incremento de confianza mayor a 1 indica que la regla de asociación explica mejor la relación entre los ítems que el azar, ratificando que en este caso la regla de asociación de más confianza entre "Yogurt" y "Leche entera" no ocurre solo por coincidencia.

```
# Reglas ordenadas por confianza
lhs                          rhs                              ... lift
{yogurt}                  => {whole milk}                     ... 1.571735
{other vegetables}        => {whole milk}                     ... 1.513634
{rolls/buns}              => {whole milk}                     ... 1.205032
{whole milk}              => {other vegetables}               ... 1.513634
...
```

Por último, si bien la validación de los modelos puede llevarse a cabo por métodos analíticos, es recomendable que sea un experto en el dominio de la aplicación quien determine si el modelo tiene sentido en el contexto particular. Si el modelo cumple con su cometido y tiene capacidades explicativas o predictivas, de acuerdo con lo esperado por el proyecto de análisis de datos, puede procederse a la última etapa, sea esta un informe, un software en producción u otra herramienta de apoyo a la toma de decisiones que pueda generar la transformación buscada por la organización.

Referencias bibliográficas

[1] R. Kabacoff, *R in Action*. Second Edition, Manning Publications Co., 2015. [En línea]. Disponible en: https://www.manning.com/books/r-in-action-second-edition

[2] E. Mayor, *Learning Predictive Analytics with R*. Packt Publishing, 2015. [En línea]. Disponible en: https://www.amazon.in/Learning-Predictive-Analytics-Eric-Mayor/dp/1782169350

[3] J. Ledolter, *Data Mining and Business Analytics with R*. John Wiley Sons, 2013. [En línea]. Disponible en: http://bit.ly/2XQyszd

[4] N. Zumel and J. Mount, *Practical Data Science with R*, Manning Publications Co., 2014. [En línea]. Disponible en: https://www.manning.com/books/practical-data-science-with-r

[5] M. Hasler, C. Buchta, B. Gruen, K. Hornik, I. Johnson and C. Borgelt. *arules: Mining Association Rules and Frequent Itemsets*, R package version 1.6-1, 2016. Disponible en [En línea]. Disponible en: https://www.rdocumentation.org/packages/arules/versions/1.6-3

Epílogo

Este libro comenzó a escribirse como un esfuerzo enciclopédico por obtener un compilado de técnicas para análisis de datos utilizando la plataforma R. La idea original era —y a decir verdad sigue siendo— resumir en un solo libro temas académicos y de cursos de formación en la *ciencia de datos*. La necesidad de un libro con enfoque práctico y en idioma español, orientado a técnicas de análisis de datos y a la plataforma R, es una herramienta de gran valor para aquellas organizaciones que deseen emprender proyectos de ciencia de datos y no cuenten con experiencia previa en estos aspectos.

Sin embargo, el objetivo inicial fue desbordado durante la escritura del libro, reduciéndose a tratar temas introductorios, pero interesantes, que deben estar contenidos, de forma mandatoria en un texto de análisis de datos en R. La reducción en el alcance del proyecto original no generó sinsabor, al contrario, amplió las expectativas intelectuales sobre la ciencia de datos, entendiendo aún más su potencial de aplicaciones, su riqueza conceptual y su amplitud en los diferentes campos del conocimiento que integra. Espero que el lector, después de leer el libro, tenga posibilidad de explorar la mayor cantidad de temáticas en la ciencia de datos.

La hoja de ruta para entender las diferentes áreas de la ciencia de datos es amplia y compleja y depende más de los distintos intereses intelectuales o de aplicación que tenga el lector, pero no quiero dejar pasar la oportunidad de exponer un comentario sobre algunas áreas llamativas en la ciencia de datos, a saber:

- Minería de texto y procesamiento de lenguaje natural. Campo de la inteligencia artificial, la estadística y la computación, cuyo objetivo es lograr que sistemas de cómputo entiendan el lenguaje natural hablado y escrito de forma similar a como lo hacen los seres humanos. Tiene interesantes aplicaciones en el *análisis de sentimientos* —o minería de opiniones—, sistema de ayuda y búsqueda (como Siri, de la empresa Apple), y automatización de procesos y procesamiento de documentos. Se ha logrado un importante avance en este aspecto, pero en lenguaje español todavía hay retos por resolver.

- Modelos estadísticos avanzados. Siendo la estadística un campo del conocimiento maduro, no es de sorprender que su cuerpo de conocimiento sea extenso y profundo. Temáticas relevantes que deberían ser parte de los conocimientos y las habilidades de un científico de datos, son el análisis multivariado, las técnicas bayesianas, la teoría estadística, el diseño de experimentos y las series de tiempo. Hoy por hoy, este último tema tiene aplicaciones con potencial económico prometedor, especialmente en las finanzas.
- Sistemas de recomendación. Herramientas de software que proveen sugerencias de ítems (información o productos) de utilidad para un usuario. Aplicaciones actuales de los sistemas de recomendación pueden encontrarse en el comercio electrónico, donde se recomiendan nuevos ítems de compra con base en lo que otros compraron o ítems que por su propio contenido se pueden considerar similares. Otras aplicaciones pueden encontrarse en el campo del entretenimiento: recomendaciones de videos, noticias, canciones u otro material multimedia. También se encuentran aplicaciones en la gestión del talento humano, donde se recomienda a los individuos empresas o trabajos a los cuales aplicar y a las empresas individuos a contratar, o incluso en redes sociales, al sugerir amigos a contactar o personas o empresas a seguir. Su aplicación práctica hace de este un tema obligado dentro de las áreas que un científico de datos debe dominar.
- Sistemas intensivos en datos o *Big Data*. En este libro se introdujo lo referente a *Big Data*, reconociendo que un tratamiento detallado del tema puede tomar varios volúmenes. El estado del arte en esta temática avanza, sin contar las aplicaciones que hacen uso de estas técnicas que ya se encuentran en el mercado. De interés en el estudio de *Big Data* son los sistemas de *streaming* o de procesamiento en tiempo real, las herramientas y modelos de programación del ecosistema del proyecto Apache (Hadoop, Spark, Storm, Hive, Pig, Flink) y en la infraestructura de tecnología y la arquitectura de un sistema intensivo en datos.
- Visualización de datos. Tema relevante, no solo desde el punto de vista técnico sino desde lo organizacional, como es la capacidad para generar visualizaciones estéticamente interesantes, informativas, precisas y oportunas, lo cual sigue siendo necesidad imperiosa.
- Administración de proyectos de datos. Sin entender a profundidad cómo gerenciar un proyecto complejo de datos no se llevarán a buen término aquellas iniciativas que aprovechan el potencial de la ciencia de datos en las organizaciones. Justamente por su naturaleza interdisciplinaria, una buena gerencia de proyectos es indispensable para que los equipos con conocimientos en las diferentes disciplinas involucradas en ciencia de datos logren esa sinergia necesaria y sean productivos.

- Aplicaciones en mercadeo, finanzas, educación, salud y ciencias sociales. Las aplicaciones de la ciencia de datos son cada vez más interesantes, en cuanto se generan nuevos conocimientos y productos intensivos en datos. El listado de campos de aplicación crece cada día más.
- Aprendizaje de máquina y aprendizaje profundo. En este libro se introduce el aprendizaje de máquina, en especial los tópicos relacionados con Aprendizaje Profundo o *Deep Learning*, un redescubrimiento de las redes neuronales artificiales que, aprovechando el poder de cómputo moderno, han generado impresionantes aplicaciones en reconocimiento de patrones, vehículos autónomos y análisis de datos.
- Otras plataformas. En este libro se trabajó con la plataforma R, pero existen en el mercado numerosas y poderosas alternativas, tanto propietarias como de código abierto. Es especialmente relevante el ecosistema Python, la herramienta MATLAB, Talend, Tableu o la plataforma para analítica KNIME, sin contar con las alternativas que la *computación en la nube* tiene para ofrecer. Casi que cualquier proveedor de servicios de nube (Google, IBM, Amazon, Microsoft) tiene un ecosistema extenso para las necesidades de análisis de datos de las organizaciones. El listado no es exhaustivo, pues solo es un abrebocas de lo interesante que resulta seguir una carrera en ciencia de datos. Por supuesto, toda aventura trae consigo un reto, que en la ciencia de datos no es menor. Espero que *Introducción a la ciencia de datos en R* sea ese primer peldaño que logre que el lector se sienta en confianza para continuar el arduo camino de convertirse en un científico de datos, ello sería suficiente para sentir que hemos contribuido al entendimiento del área y todo el potencial que tiene para ofrecer.

Autor

José Nelson Pérez Castillo

Profesor titular de carrera adscrito a la Facultad de Ingeniería de la Universidad Distrital Francisco José de Caldas desde 1995. Doctor en Informática de la Universidad de Oviedo. Magíster en Teleinformática de la Universidad Distrital Francisco José de Caldas. Especialista en Cartografía, Sistemas de Información Geográfica y Teledetección de la Universidad de Alcalá de Henares. Especialista en Sistemas de Información Geográfica de la Universidad Distrital Francisco José de Caldas e Ingeniero de Sistemas egresado de la misma institución. Investigador Senior, Ministerio de Ciencia, Tecnología e Innovación. Director del grupo internacional de investigación en Informática, Comunicaciones y Gestión del Conocimiento (Gicoge).

Este libro se
terminó de imprimir
en mayo de 2020
en la Editorial UD
Bogotá, Colombia