Gestión y comprensión de los datos

Un componente clave inicial de cualquier proyecto de aprendizaje automático implica la gestión y la comprensión de los datos. Aunque esto puede no ser tan gratificante como la creación e implementación de modelos (las etapas en las que se empiezan a ver los frutos del trabajo), no es prudente ignorar este importante trabajo preparatorio.

Cualquier algoritmo de aprendizaje es tan bueno como sus datos de entrada y, en muchos casos, estos son complejos, desordenados y están distribuidos en múltiples fuentes y formatos.
Debido a esta complejidad, a menudo la mayor parte del esfuerzo invertido en proyectos de aprendizaje automático se destina a la preparación y exploración de datos.

Este documento aborda la preparación de datos de tres maneras. La primera sección analiza las estructuras de datos básicas que R utiliza para almacenar datos. Te familiarizarás con estas estructuras a medida que crees y manipules conjuntos de datos. La segunda sección es práctica, ya que cubre varias funciones que se utilizan para introducir y extraer datos de R. En la tercera sección, se ilustran métodos para comprender los datos mientras se explora un conjunto de datos del mundo real.

Al finalizar este documento, comprenderás:

- Cómo usar las estructuras de datos básicas de R para almacenar y manipular valores
- Funciones simples para obtener datos en R desde formatos de origen comunes
- Métodos típicos para comprender y visualizar datos complejos

Las formas en que R administra los datos dictarán las formas en que debes trabajar con los datos, por lo que es útil conocer las estructuras de datos de R antes de pasar directamente a la sección sobre preparación de datos.

Sin embargo, si ya estás familiarizado con la programación en R, no dudes en pasar directamente al documento sobre preprocesamiento de datos.

Estructuras de datos de R

Existen numerosos tipos de estructuras de datos en los lenguajes de programación, cada uno con fortalezas y debilidades adecuadas para tareas específicas. Dado que R es un lenguaje de

programación ampliamente utilizado para el análisis de datos estadísticos, las estructuras de datos que utiliza se diseñaron teniendo en cuenta este tipo de trabajo.

Las estructuras de datos de R que se utilizan con mayor frecuencia en el aprendizaje automático son vectores, factores, listas, matrices y frames de datos. Cada uno está diseñado para una tarea de administración de datos específica, lo que hace que sea importante comprender cómo interactuarán en tu proyecto de R. En las secciones siguientes, revisaremos sus similitudes y diferencias.

Vectores

La estructura de datos fundamental de R es el **vector**, que almacena un conjunto ordenado de valores llamados **elementos**. Un vector puede contener cualquier cantidad de elementos. Sin embargo, todos sus elementos deben ser del mismo tipo; por ejemplo, un vector no puede contener números y texto. Para determinar el tipo de vector v, utiliza el comando typeof(v).

Se utilizan varios tipos de vectores en el aprendizaje automático: integer (números sin decimales), double (números con decimales), character (datos de texto) y logical (valores VERDADEROS o FALSO). También hay dos valores especiales: NA, que indica un valor faltante, y NULL, que se utiliza para indicar la ausencia de cualquier valor. Aunque estos dos pueden parecer sinónimos, en realidad son ligeramente diferentes. El valor NA es un marcador de posición para otra cosa y, por lo tanto, tiene una longitud de uno, mientras que el valor NULL está realmente vacío y tiene una longitud de cero.

Algunas funciones de R reportarán ambos vectores, los enteros y los dobles como numéricos, mientras que otras distinguirán entre los dos. Como resultado, aunque todos los vectores dobles son numéricos, no todos los vectores numéricos son de tipo doble.

Es tedioso ingresar grandes cantidades de datos a mano, pero se pueden crear vectores simples utilizando la función de combinación c(). También se puede asignar un nombre al vector utilizando el operador de flecha <-. Este es el operador de asignación de R, que se utiliza de manera muy similar al operador de asignación = en muchos otros lenguajes de programación.

Por ejemplo, construyamos un conjunto de vectores que contengan datos sobre tres pacientes médicos.
Crearemos un vector de caracteres llamado subject_name para almacenar los nombres de los tres pacientes, un vector numérico llamado temperature para almacenar la temperatura corporal de cada paciente en grados Fahrenheit y un vector lógico llamado flu_status para almacenar el diagnóstico de cada paciente (TRUE si tiene influenza, FALSE en caso contrario). Como se muestra en el código siguiente, los tres vectores son:

```
> subject_name <- c("John Doe", "Jane Doe", "Steve Graves")
> temperature <- c(98.1, 98.6, 101.4)
> flu_status <- c(FALSE, FALSE, TRUE)
```

Los valores almacenados en los vectores R conservan su orden. Por lo tanto, se puede acceder a los datos de cada paciente utilizando su posición en el conjunto, comenzando en 1, y luego proporcionando este número dentro de corchetes (es decir, [y]) después del nombre del vector.
Por ejemplo, para obtener el valor de temperatura de la paciente Jane Doe, la segunda paciente, simplemente escribe:

```
> temperature[2]
[1] 98.6
```

R ofrece una variedad de métodos para extraer datos de vectores. Se puede obtener un rango de valores utilizando el operador de dos puntos. Por ejemplo, para obtener la temperatura corporal del segundo y tercer paciente, escribe:

```
> temperature[2:3]
[1]  98.6 101.4
```

Los elementos se pueden excluir especificando un número de elemento negativo. Para excluir los datos de temperatura del segundo paciente, escribe:

```
> temperature[-2]
[1]  98.1 101.4
```

A veces también es útil especificar un vector lógico que indique si se debe incluir o no cada elemento. Por ejemplo, para incluir las dos primeras lecturas de temperatura pero excluir la tercera, escribe:

```
> temperature[c(TRUE, TRUE, FALSE)]
[1] 98.1 98.6
```

Como verás en breve, el vector proporciona la base para muchas otras estructuras de datos de R. Por lo tanto, conocer las diversas operaciones vectoriales es crucial para trabajar con datos en R.

Factores

Si recuerdas el primer documento, Introducción al aprendizaje automático, las características nominales representan una característica con categorías de valores. Aunque es posible utilizar

un vector de caracteres para almacenar datos nominales, R proporciona una estructura de datos específicamente para este propósito.

Un **factor** es un caso especial de vector que se utiliza únicamente para representar variables categóricas u ordinales. En el conjunto de datos médicos que estamos creando, podríamos utilizar un factor para representar el género porque utiliza dos categorías: masculino y femenino.

¿Por qué utilizar factores en lugar de vectores de caracteres? Una ventaja de los factores es que las etiquetas de categoría se almacenan solo una vez. En lugar de almacenar HOMBRE, HOMBRE, FEMENINO, la computadora puede almacenar 1, 1, 2, lo que puede reducir la memoria necesaria para almacenar los valores. Además, muchos algoritmos de aprendizaje automático tratan las características nominales y numéricas de manera diferente. Codificar las variables categóricas como factores garantiza que R manejará los datos categóricos de manera adecuada.

No se debe utilizar un factor para vectores de caracteres que no sean verdaderamente categóricos. En particular, si un vector almacena principalmente valores únicos, como nombres o códigos de identificación, mantenlo como un vector de caracteres.

Para crear un factor a partir de un vector de caracteres, simplemente aplica la función factor(). Por ejemplo:

```
> gender <- factor(c("MALE", "FEMALE", "MALE"))
> gender
[1] MALE   FEMALE MALE
Levels: FEMALE MALE
```

Observa que cuando se mostró el factor de género, R imprimió información adicional sobre sus niveles. Los niveles comprenden el conjunto de posibles categorías que podría tomar el factor, en este caso, HOMBRE o FEMENINO.

Cuando creamos factores, podemos agregar niveles adicionales que pueden no aparecer en los datos originales. Supongamos que creamos otro factor para el tipo de sangre, como se muestra en el siguiente ejemplo:

```
> blood <- factor(c("O", "AB", "A"),
+   levels = c("A", "B", "AB", "O"))
> blood
[1] O  AB A
Levels: A B AB O
```

Cuando definimos el factor sangre, especificamos un vector adicional de cuatro tipos de sangre posibles utilizando el parámetro niveles. Como resultado, aunque nuestros datos incluyen solo los tipos de sangre O, AB y A, los cuatro tipos se conservan con el factor sangre, como se muestra en el resultado.

El almacenamiento del nivel adicional permite la posibilidad de agregar pacientes con el otro tipo de sangre en el futuro. También garantiza que, si creáramos una tabla de tipos de sangre, sabríamos que existe el tipo B, a pesar de que no se encuentra en nuestros datos iniciales.

La estructura de datos del factor también nos permite incluir información sobre el orden de las categorías de una variable nominal, lo que proporciona un método para crear características ordinales.

Por ejemplo, supongamos que tenemos datos sobre la gravedad de los síntomas del paciente, codificados en orden creciente de gravedad desde leve a moderado y grave. Indicamos la presencia de datos ordinales proporcionando los niveles del factor en el orden deseado, enumerados en orden ascendente del más bajo al más alto, y configurando el parámetro ordenado en VERDADERO como se muestra:

```
> symptoms <- factor(c("SEVERE", "MILD", "MODERATE"),
+   levels = c("MILD", "MODERATE", "SEVERE"),
+   ordered = TRUE)
```

El factor de síntomas resultante ahora incluye información sobre el orden solicitado. A diferencia de nuestros factores anteriores, los niveles de este factor están separados por símbolos < para indicar la presencia de un orden secuencial de LEVE a GRAVE:

```
> symptoms
[1] SEVERE   MILD     MODERATE
Levels: MILD < MODERATE < SEVERE
```

Una característica útil de los factores ordenados es que las pruebas lógicas funcionan como se esperaría. Por ejemplo, podemos probar si los síntomas de cada paciente son más severos que moderados:

```
> symptoms > "MODERATE"
[1]  TRUE FALSE FALSE
```

Los algoritmos de aprendizaje automático capaces de modelar datos ordinales esperarán factores ordenados, así que asegúrese de codificar sus datos en consecuencia.

Listas

Una **lista** es una estructura de datos, muy similar a un vector, ya que se utiliza para almacenar un conjunto ordenado de elementos. Sin embargo, mientras que un vector requiere que todos sus elementos sean del mismo tipo, una lista permite recopilar diferentes tipos de datos R. Debido a esta flexibilidad, las listas se utilizan a menudo para almacenar varios tipos de datos de entrada y salida y conjuntos de parámetros de configuración para modelos de aprendizaje automático.

Para ilustrar las listas, considera el conjunto de datos de pacientes médicos que hemos estado construyendo, con datos de tres pacientes almacenados en seis vectores. Si quisiéramos mostrar todos los datos del primer paciente, tendríamos que ingresar cinco comandos R:

```
> subject_name[1]
[1] "John Doe"
> temperature[1]
[1] 98.1
> flu_status[1]
[1] FALSE
> gender[1]
[1] MALE
Levels: FEMALE MALE
> blood[1]
[1] O
Levels: A B AB O
>  symptoms[1]
[1] SEVERE
Levels: MILD < MODERATE < SEVERE
```

Si esperamos examinar los datos del paciente nuevamente en el futuro, en lugar de volver a escribir estos comandos, una lista nos permite agrupar todos los valores en un objeto que podemos usar repetidamente.

De manera similar a la creación de un vector con c(), se crea una lista utilizando la función list() como se muestra en el siguiente ejemplo.

Una diferencia notable es que cuando se construye una lista, se debe asignar un nombre a cada componente de la secuencia. Los nombres no son estrictamente necesarios, pero permiten acceder a los valores más adelante por nombre en lugar de por posición numerada.

Para crear una lista con componentes nombrados para todos los datos del primer paciente, escribe lo siguiente:

```
> subject1 <- list(fullname = subject_name[1],
+   temperature = temperature[1],
+   flu_status = flu_status[1],
+   gender = gender[1],
+   blood = blood[1],
+   symptoms = symptoms[1])
```

Los datos de este paciente ahora se recopilan en la lista subject1:

```
> subject1
$fullname
[1] "John Doe"

$temperature
[1] 98.1

$flu_status
[1] FALSE

$gender
[1] MALE
Levels: FEMALE MALE

$blood
[1] O
Levels: A B AB O

$symptoms
[1] SEVERE
Levels: MILD < MODERATE < SEVERE
```

Observa que los valores están etiquetados con los nombres que especificamos en el comando anterior. Como una lista conserva el orden como un vector, se puede acceder a sus componentes mediante posiciones numéricas, como se muestra aquí para el valor de temperatura:

```
> subject1[2]
$temperature
[1] 98.1
```

El resultado de usar operadores de estilo vectorial en un objeto de lista es otro objeto de lista, que es un subconjunto de la lista original. Por ejemplo, el código anterior devolvió una lista

con un solo componente de temperatura. Para devolver en cambio un solo elemento de lista en su tipo de datos nativo, use corchetes dobles ([[y]]) al seleccionar el componente de lista. Por ejemplo, el siguiente comando devuelve un vector numérico de longitud uno:

```
> subject1[[2]]
[1] 98.1
```

Para mayor claridad, a menudo es mejor acceder a los componentes de la lista por nombre, agregando un $ y el nombre del componente al nombre de la lista de la siguiente manera:

```
> subject1$temperature
[1] 98.1
```

Al igual que la notación de corchetes dobles, esto devuelve el componente de lista en su tipo de datos nativo (en este caso, un vector numérico de longitud uno).

El acceso al valor por nombre también garantiza que se recupere el elemento correcto incluso si el orden de los elementos de la lista se cambia más adelante.

Es posible obtener varios elementos de lista especificando un vector de nombres. A continuación, se devuelve un subconjunto de la lista subject1, que contiene solo los componentes temperature y flu_status:

```
> subject1[c("temperature", "flu_status")]
$temperature
[1] 98.1

$flu_status
[1] FALSE
```

Se pueden construir conjuntos de datos completos utilizando listas y listas de listas. Por ejemplo, puedes considerar la posibilidad de crear una lista subject2 y subject3, y agruparlas en un objeto de lista llamado pt_data. Sin embargo, construir un conjunto de datos de esta manera es lo suficientemente común como para que R proporcione una estructura de datos especializada específicamente para esta tarea.

Frames de datos

La estructura de datos R más importante que se utiliza en el aprendizaje automático es, **el frame de datos**, una estructura análoga a una hoja de cálculo o una base de datos en el sentido de que tiene filas y columnas de datos. En términos de R, un frame de datos puede entenderse como una lista de vectores o factores, cada uno de los cuales tiene exactamente la misma

cantidad de valores. Ahora bien, como el frame de datos es literalmente una lista de objetos de tipo vectorial, combina aspectos tanto de vectores como de listas.

Creemos un frame de datos para nuestro conjunto de datos de pacientes. Utilizando los vectores de datos de pacientes que creamos anteriormente, la función data.frame() los combina en un frame de datos:

```
> pt_data <- data.frame(subject_name, temperature,
+   flu_status, gender, blood, symptoms,
+   stringsAsFactors = FALSE)
```

Puede que notes algo nuevo en el código anterior. Incluimos un parámetro adicional: stringsAsFactors = FALSE. Si no especificamos esta opción, R convertirá automáticamente cada vector de caracteres en un factor.

Esta función es útil en ocasiones, pero también a veces injustificada. Aquí, por ejemplo, el campo subject_name definitivamente no es un dato categórico, ya que los nombres no son categorías de valores. Por lo tanto, configurar la opción stringsAsFactors en FALSE nos permite convertir vectores de caracteres en factores solo cuando tenga sentido para el proyecto.

Cuando mostramos el frame de datos pt_data, vemos que la estructura es bastante diferente de las estructuras de datos con las que trabajamos anteriormente:

```
> pt_data
  subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
2     Jane Doe        98.6      FALSE FEMALE    AB     MILD
3 Steve Graves       101.4       TRUE   MALE     A MODERATE
```

En comparación con los vectores, factores y listas unidimensionales, un frame de datos tiene dos dimensiones y se muestra en formato de matriz. Este frame de datos en particular tiene una columna para cada vector de datos del paciente y una fila para cada paciente. En términos de aprendizaje automático, las columnas del frame de datos son las características o atributos y las filas son los ejemplos.

Para extraer columnas completas (vectores) de datos, podemos aprovechar el hecho de que un frame de datos es simplemente una lista de vectores. De manera similar a las listas, la forma más directa de extraer un solo elemento es haciendo referencia a él por su nombre. Por ejemplo, para obtener el vector subject_name, escribe:

```
> pt_data$subject_name
[1] "John Doe"     "Jane Doe"     "Steve Graves"
```

También de manera similar a las listas, se puede utilizar un vector de nombres para extraer varias columnas de un frame de datos:

```
> pt_data[c("temperature", "flu_status")]
    temperature   flu_status
1        98.1         FALSE
2        98.6         FALSE
3       101.4         TRUE
```

Cuando solicitamos columnas en el frame de datos por nombre, el resultado es un frame de datos que contiene todas las filas de datos para las columnas especificadas. El comando pt_data[2:3] también extraerá las columnas temperature y flu_status. Sin embargo, hacer referencia a las columnas por nombre da como resultado un código R claro y fácil de mantener que no se romperá si el frame de datos se reordena más tarde.

Para extraer valores específicos del frame de datos, se utilizan métodos como los que se utilizan para acceder a valores en vectores. Sin embargo, existe una distinción importante: debido a que el frame de datos es bidimensional, se deben especificar tanto las filas como las columnas deseadas. Las filas se especifican primero, seguidas de una coma, seguidas de las columnas en un formato como este: [filas, columnas]. Al igual que con los vectores, las filas y las columnas se cuentan comenzando en uno.

Por ejemplo, para extraer el valor de la primera fila y la segunda columna del frame de datos del paciente, utiliza el siguiente comando:

```
> pt_data[1, 2]
[1] 98.1
```

Si deseas más de una sola fila o columna de datos, especifica vectores que indiquen las filas y columnas deseadas. La siguiente instrucción extraerá datos de la primera y la tercera fila y de la segunda y la cuarta columna:

```
> pt_data[c(1, 3), c(2, 4)]
  Temperature  gender
1        98.1    MALE
3       101.4    MALE
```

Para hacer referencia a cada fila o columna, simplemente deja la parte de la fila o columna en blanco. Por ejemplo, para extraer todas las filas de la primera columna:

```
> pt_data[, 1]
[1] "John Doe"     "Jane Doe"     "Steve Graves"
```

Para extraer todas las columnas de la primera fila:

```
> pt_data[1, ]
  subject_name   temperature   flu_status    gender    blood    symptoms
1     John Doe         98.1          FALSE      MALE      O       SEVERE
```

Y para extraer todo:

```
> pt_data[ , ]
```

```
   subject_name temperature flu_status gender blood symptoms
1      John Doe        98.1      FALSE   MALE     O   SEVERE
2      Jane Doe        98.6      FALSE FEMALE    AB     MILD
3 Steve Graves       101.4       TRUE   MALE     A MODERATE
```

Por supuesto, es mejor acceder a las columnas por nombre en lugar de por posición, y se pueden utilizar signos negativos para excluir filas o columnas de datos. Por lo tanto, la salida del comando:

```
> pt_data[c(1, 3), c("temperature", "gender")]
  Temperatura  gender
1        98.1    MALE
3       101.4    MALE
```

es equivalente a:

```
> pt_data[-2, c(-1, -3, -5, -6)]
  temperature   gender
1        98.1    MALE
3       101.4    MALE
```

A veces es necesario crear nuevas columnas en los frames de datos, tal vez, por ejemplo, como una función de las columnas existentes. Por ejemplo, es posible que necesitemos convertir las lecturas de temperatura Fahrenheit en el frame de datos del paciente a la escala Celsius.
Para ello, simplemente utilizamos el operador de asignación para asignar el resultado del cálculo de conversión a un nuevo nombre de columna de la siguiente manera:

```
> pt_data$temp_c <- (pt_data$temperature - 32) * (5 / 9)
```

Para confirmar que el cálculo funcionó, comparemos la nueva columna temp_c en escala Celsius con la columna de temperatura anterior en escala Fahrenheit:

```
> pt_data[c("temperature", "temp_c")]
  temperature  temp_c
1        98.1 36.72222
2        98.6 37.00000
3       101.4 38.55556
```

Al verlas una al lado de la otra, podemos confirmar que el cálculo funcionó correctamente. Para familiarizarse más con los frames de datos, intenta practicar operaciones similares con el conjunto de datos del paciente o, mejor aún, utiliza datos de uno de tus propios proyectos. Este tipo de operaciones son cruciales para gran parte del trabajo que realizaremos en los próximos temas.

Matrices y arreglos

Además de los frames de datos, R proporciona otras estructuras que almacenan valores en forma de tabla. Una **matriz** es una estructura de datos que representa una tabla bidimensional con filas y columnas de datos. Al igual que los vectores, las matrices de R pueden contener solo un tipo de datos, aunque se utilizan con mayor frecuencia para operaciones matemáticas y, por lo tanto, normalmente almacenan solo números.

Para crear una matriz, simplemente proporciona un vector de datos a la función matrix(), junto con un parámetro que especifique el número de filas (nrow) o el número de columnas (ncol). Por ejemplo, para crear una matriz 2 × 2 que almacene los números del uno al cuatro, podemos usar el parámetro nrow para solicitar que los datos se dividan en dos filas:

```
> m <- matrix(c(1, 2, 3, 4), nrow = 2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Esto es equivalente a la matriz producida utilizando ncol = 2:

```
> m <- matrix(c(1, 2, 3, 4), ncol = 2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Notarás que R cargó la primera columna de la matriz primero antes de cargar la segunda columna. Esto se llama orden principal de columnas, que es el método predeterminado de R para cargar matrices.

Para anular esta configuración predeterminada y cargar una matriz por filas, configura el parámetro byrow = TRUE al crear la matriz.

Para ilustrar esto más detalladamente, veamos qué sucede si agregamos más valores a la matriz.
Con seis valores, al solicitar dos filas se crea una matriz con tres columnas:

```
> m <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Al solicitar dos columnas se crea una matriz con tres filas:

```
> m <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
> m
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Al igual que con los frames de datos, los valores de las matrices se pueden extraer utilizando la notación [fila, columna]. Por ejemplo, m[1, 1] devolverá el valor 1, mientras que m[3, 2] extraerá 6 de la matriz m. Además, se pueden solicitar filas o columnas completas:

```
> m[1, ]
[1] 1 4
> m[, 1]
[1] 1 2 3
```

Estrechamente relacionada con la estructura de la matriz se encuentra el **arreglo**, que es una tabla multidimensional de datos. Mientras que una matriz tiene filas y columnas de valores, un arreglo tiene filas, columnas y una serie de capas adicionales de valores.

Aunque ocasionalmente utilizaremos matrices en temas posteriores, el uso de arreglos es innecesario dentro del alcance de este curso.

Administración de datos con R

Uno de los desafíos que se enfrentan al trabajar con conjuntos de datos masivos implica recopilar, preparar y administrar datos de una variedad de fuentes. Aunque cubriremos la preparación, limpieza y administración de datos en profundidad trabajando en tareas de aprendizaje automático del mundo real en temas posteriores, esta sección destaca la funcionalidad básica para ingresar y extraer datos de R.

Guardar, cargar y eliminar estructuras de datos de R

Cuando hayas pasado mucho tiempo obteniendo un frame de datos en la forma deseada, no deberías necesitar recrear tu trabajo cada vez que reinicies tu sesión de R. Para guardar una estructura de datos en un archivo que se pueda volver a cargar más tarde o transferir a otro sistema, usa la función save(). La función save() escribe una o más estructuras de datos de R en la ubicación especificada por el parámetro de archivo. Los archivos de datos de R tienen una extensión .RData.

Supongamos que tienes tres objetos llamados *x*, *y* y *z* que te gustaría guardar en un archivo permanente. Independientemente de si son vectores, factores, listas o frames de datos, se pueden guardar en un archivo llamado mydata.RData usando el siguiente comando (no lo ejecutes!):

```
> save(x, y, z, file = "mydata.RData")
```

El comando load() puede recrear cualquier estructura de datos que se haya guardado en un archivo .RData. Para cargar el archivo mydata.RData creado en el código anterior, simplemente escribe (tampoco lo ejecutes!):

```
> load("mydata.RData")
```

Esto recreará las estructuras de datos *x*, *y*, *z* en tu entorno R.

> ¡Ten cuidado con lo que estás cargando! Todas las estructuras de datos almacenadas en el archivo que estás importando con el comando load() se agregarán a tu espacio de trabajo, incluso si sobrescriben algo más en lo que estás trabajando.

Si necesitas terminar tu sesión de R rápidamente, el comando save.image() escribirá toda tu sesión en un archivo llamado simplemente .RData. De manera predeterminada, R buscará este archivo la próxima vez que inicies R y tu sesión se volverá a crear tal como la dejaste.

Después de haber estado trabajando en una sesión de R durante algún tiempo, es posible que hayas acumulado varias estructuras de datos. La función de listado ls() devuelve un vector de todas las estructuras de datos que se encuentran actualmente en la memoria. Por ejemplo, si has estado siguiendo el código de este documento, la función ls() devuelve lo siguiente:

```
> ls()
[1] "blood" "flu_status" "gender" "m"
[5] "subject_name" "subject1" "symptoms"
[9] "temperature"
```

R borra automáticamente todas las estructuras de datos de la memoria al salir de la sesión, pero para objetos grandes, es posible que quieras liberar la memoria antes. La función de eliminación rm() se puede utilizar para este propósito. Por ejemplo, para eliminar los objetos m y subject1, simplemente escriba:

```
> rm(m, subject1)
```

La función rm() también puede recibir un vector de caracteres de los nombres de los objetos que se eliminarán. Esto funciona con la función ls() para borrar toda la sesión R:

```
> rm(list = ls())
```

Ten mucho cuidado al ejecutar el código anterior, ya que no se te solicitará nada antes de eliminar los objetos.

Importación y guardado de datos de archivos CSV

Es muy común que los conjuntos de datos públicos se almacenen en archivos de texto. Los archivos de texto se pueden leer en prácticamente cualquier computadora o sistema operativo, lo que hace que el formato sea casi universal. También se pueden exportar e importar desde y hacia programas como Microsoft Excel, lo que proporciona una forma rápida y sencilla de trabajar con datos de hojas de cálculo.

Un archivo de datos tabular (como en "tabla") está estructurado en forma de matriz, de modo que cada línea de texto refleja un ejemplo y cada ejemplo tiene la misma cantidad de características.
Los valores de las características en cada línea están separados por un símbolo predefinido conocido como **delimitador**. A menudo, la primera línea de un archivo de datos tabular enumera los nombres de las columnas de datos. Esto se denomina **línea de encabezado**.

Quizás el formato de archivo de texto tabular más común es el archivo de valores separados por comas (CSV), que, como sugiere el nombre, utiliza la coma como delimitador. Los archivos CSV se pueden importar y exportar desde muchas aplicaciones comunes. Un archivo CSV que represente el conjunto de datos médicos creado previamente se podría almacenar como:

```
subject_name,temperature,flu_status,gender,blood_type
John Doe,98.1,FALSE,MALE,O
Jane Doe,98.6,FALSE,FEMALE,AB
Steve Graves,101.4,TRUE, MALE,A
```

Dado un archivo de datos de pacientes llamado pt_data.csv ubicado en el directorio de trabajo de R, la función read.csv() se puede utilizar de la siguiente manera para cargar el archivo en R:

```
> pt_data <- read.csv("pt_data.csv", stringsAsFactors = FALSE)
```

Esto leerá el archivo CSV en un frame de datos titulado pt_data. Tal como lo habíamos hecho anteriormente al construir un frame de datos, necesitamos utilizar el parámetro stringsAsFactors = FALSE para evitar que R convierta todas las variables de texto en factores. A menos que estés seguro de que cada columna del archivo CSV es realmente un factor, es mejor dejar que este paso lo realices tú, no R.

> Si tu conjunto de datos se encuentra fuera del directorio de trabajo de R, se puede utilizar la ruta completa al archivo CSV (por ejemplo, "/path/to/mydata.csv") al llamar a la función read.csv().

De manera predeterminada, R supone que el archivo CSV incluye una línea de encabezado que enumera los nombres de las características del conjunto de datos. Si un archivo CSV no tiene un encabezado, especifica la opción header = FALSE como se muestra en el siguiente comando y R asignará nombres de características predeterminados numerándolos como V1, V2, etc.:

```
> mydata <- read.csv("mydata.csv", stringsAsFactors = FALSE,
+  header = FALSE)
```

La función read.csv() es un caso especial de la función read.table(), que puede leer datos tabulares en muchas formas diferentes, incluidos otros formatos delimitados como **valores separados por tabulaciones** (TSV, Tab-separated values). Para obtener información más detallada sobre la familia de funciones read.table(), consulta la página de ayuda de R utilizando el comando ?read.table.

Para guardar un frame de datos en un archivo CSV, utiliza la función write.csv(). Si tu frame de datos se llama pt_data, simplemente ingresa:

```
> write.csv(pt_data, file = "pt_data.csv", row.names = FALSE)
```

Esto escribirá un archivo CSV con el nombre pt_data.csv en la carpeta de trabajo de R.
El parámetro row.names anula la configuración predeterminada de R, que es generar los nombres de las filas en el archivo CSV. Generalmente, esta salida es innecesaria y simplemente inflará el tamaño del archivo resultante.

Exploración y comprensión de los datos

Después de recopilar datos y cargarlos en las estructuras de datos de R, el siguiente paso en el proceso de aprendizaje automático implica examinar los datos en detalle. Durante este paso, comenzarás a explorar las características y los ejemplos de los datos y te darás cuenta de las peculiaridades que hacen que tus datos sean únicos.

Cuanto mejor comprendas tus datos, mejor podrás adaptar un modelo de aprendizaje automático a tu problema de aprendizaje.

La mejor manera de aprender el proceso de exploración de datos es mediante ejemplos. En esta sección, exploraremos el conjunto de datos usedcars.csv, que contiene datos reales sobre autos usados anunciados para la venta en un sitio web popular de EE. UU. en el año 2012. El archivo se adjunta en el documento. De manera predeterminada en Windows, los archivos se guardan en la carpeta **Documentos**.

Como el conjunto de datos se almacena en formato CSV, podemos usar la función read.csv() para cargar los datos en un frame de datos de R:

```
> usedcars <- read.csv("usedcars.csv", stringsAsFactors = FALSE)
```

Dado el frame de datos usedcars, ahora asumiremos el papel de un científico de datos que tiene la tarea de comprender los datos de autos usados. Aunque la exploración de datos es un proceso fluido, los pasos pueden imaginarse como una especie de investigación en la que se responden preguntas sobre los datos.

Las preguntas exactas pueden variar según el proyecto, pero los tipos de preguntas son siempre similares. Debería poder adaptar los pasos básicos de esta investigación a cualquier conjunto de datos que desee, ya sea grande o pequeño.

Exploración de la estructura de los datos

Las primeras preguntas que debes plantearte en una investigación de un nuevo conjunto de datos deben ser sobre cómo está organizado el conjunto de datos. Si tienes suerte, tu fuente proporcionará un **diccionario de datos**, un documento que describe las características del conjunto de datos. En nuestro caso, los datos de los automóviles usados no vienen con esta documentación, por lo que necesitaremos crear la nuestra.

La función str() proporciona un método para mostrar la estructura de objetos R, como frames de datos, vectores o listas. Puede utilizarse para crear el esquema básico de nuestro diccionario de datos:

> str(usedcars)

```
'data.frame':   150 obs. of  6 variables:
 $ year        : int  2011 2011 2011 2011 2012 2010 2011 2010 2011 2010 ...
 $ model       : chr  "SEL" "SEL" "SEL" "SEL" ...
 $ price       : int  21992 20995 19995 17809 17500 17495 17000 16995 16995 16995 ...
 $ mileage     : int  7413 10926 7351 11613 8367 25125 27393 21026 32655 36116 ...
 $ color       : chr  "Yellow" "Gray" "Silver" "Gray" ...
 $ transmission: chr  "AUTO" "AUTO" "AUTO" "AUTO" ...
```

Para un comando tan simple, aprendemos una gran cantidad de información sobre el conjunto de datos.
La declaración 150 obs nos informa que los datos incluyen 150 observaciones, que es simplemente otra forma de decir que el conjunto de datos contiene 150 registros o ejemplos. El número de observaciones a menudo se abrevia simplemente como *n*.

Como sabemos que los datos describen automóviles usados, ahora podemos suponer que tenemos ejemplos de $n = 150$ automóviles a la venta.
La declaración de 6 variables se refiere a las seis características que se registraron en los datos.
Estas características se enumeran por nombre en líneas separadas. Al observar la línea de la característica llamada color, notamos algunos detalles adicionales:

$ color   : chr "Yellow" "Gray" "Silver" "Gray" ...

Después del nombre de la variable, la etiqueta chr nos dice que la característica es de tipo carácter.

En este conjunto de datos, tres de las variables son de tipo carácter, mientras que las otras tres se indican como int, que se refiere al tipo entero. Aunque el conjunto de datos de autos usados incluye solo variables de tipo carácter y entero, es probable que también encuentres

variables de tipo numérico o num cuando utilices datos que no sean enteros. Todos los factores se enumerarán como tipo de factor. Después del tipo de cada variable, R presenta una secuencia de los primeros valores de las características. Los valores "Yellow", "Gray", "Silver", "Gray" son los primeros cuatro valores de la característica de color.

Si aplicamos un poco de conocimiento del área temática a los nombres y valores de las características, podemos hacer algunas suposiciones sobre lo que representan las variables. La variable de año podría referirse al año en que se fabricó el vehículo o podría especificar el año en que se publicó el anuncio. Tendremos que investigar esta característica más adelante con más detalle, ya que los cuatro valores de ejemplo (2011 2011 2011 2011) podrían usarse para argumentar a favor de cualquiera de las dos posibilidades. Las variables de modelo, precio, kilometraje, color y transmisión probablemente se refieran a las características del automóvil en venta.

Aunque parece que a nuestros datos se les han asignado nombres de variables significativos, no siempre es así. A veces, los conjuntos de datos tienen características con nombres o códigos sin sentido, como V1. En estos casos, puede ser necesario realizar una investigación adicional para determinar qué representa realmente una característica. De todos modos, incluso con nombres de características útiles, siempre es prudente ser escéptico sobre las etiquetas proporcionadas. Investiguemos más.

Explorando variables numéricas

Para investigar las variables numéricas en los datos de autos usados, emplearemos un conjunto común de mediciones para describir valores conocidos como **estadísticas de resumen**. La función summary() muestra varias estadísticas de resumen comunes.
Echemos un vistazo a una sola característica, el año:

```
> summary(usedcars$year)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2000    2008    2009    2009    2010    2012
```

Ignorando el significado de los valores por ahora, el hecho de que veamos números como 2000, 2008 y 2009 nos lleva a creer que la variable del año indica el año de fabricación en lugar del año en que se publicó el anuncio, ya que sabemos que los listados de vehículos se obtuvieron en 2012.

Al proporcionar un vector de nombres de columnas, también podemos utilizar la función summary() para obtener estadísticas de resumen para varias variables numéricas al mismo tiempo:

```
> summary(usedcars[c("price", "mileage")])

     price          mileage
 Min.   : 3800   Min.   :  4867
 1st Qu.:10995   1st Qu.: 27200
 Median :13592   Median : 36385
 Mean   :12962   Mean   : 44261
 3rd Qu.:14904   3rd Qu.: 55125
 Max.   :21992   Max.   :151479
```

Las seis estadísticas de resumen que proporciona la función summary() son herramientas simples, pero poderosas para investigar datos. Se pueden dividir en dos tipos: medidas de centro y medidas de dispersión.

Medición de la tendencia central: media y mediana

Las medidas de tendencia central son una clase de estadísticas que se utilizan para identificar un valor que se encuentra en el medio de un conjunto de datos. Es muy probable que ya estés familiarizado con una medida de centro común: el promedio. En el uso común, cuando algo se considera promedio, se encuentra en algún lugar entre los extremos de la escala.

Un estudiante promedio puede tener notas que se ubiquen en el medio de las de sus compañeros de clase.
Un peso promedio no es inusualmente liviano ni pesado. En general, un artículo promedio es típico y no muy diferente de los demás en su grupo. Puedes pensar en él como un ejemplo por el cual se juzgan todos los demás.

En estadística, el promedio también se conoce como la media, que es una medida definida como la suma de todos los valores dividida por la cantidad de valores. Por ejemplo, para calcular el ingreso medio en un grupo de tres personas con ingresos de $36,000, $44,000 y $56,000, podríamos escribir:

```
> (36000 + 44000 + 56000) / 3
[1] 45333.33
```

R también proporciona una función media(), que calcula la media para un vector de números:

```
> mean(c(36000, 44000, 56000))
[1] 45333.33
```

El ingreso medio de este grupo de personas es de aproximadamente $45,333. Conceptualmente, esto se puede imaginar como el ingreso que cada persona tendría si la cantidad total de ingresos se dividiera equitativamente entre todas las personas.

Recuerda que el resultado de summary() anterior enumeraba los valores medios para las variables de precio y kilometraje. Estos valores sugieren que el coche usado típico de este conjunto de datos tenía un precio de venta de 12,962 dólares y una lectura del odómetro de 44,261. ¿Qué nos dice esto sobre nuestros datos? Podemos observar que, como el precio medio es relativamente bajo, podríamos esperar que el conjunto de datos contuviera coches de clase económica. Por supuesto, los datos también pueden incluir coches de lujo de último modelo con un alto kilometraje, pero la estadística de kilometraje medio relativamente bajo no proporciona evidencia que respalde esta hipótesis. Por otro lado, tampoco proporciona evidencia para ignorar la posibilidad. Tendremos que tener esto en cuenta a medida que examinemos los datos más a fondo.

Aunque la media es, con diferencia, la estadística más citada para medir el centro de un conjunto de datos, no siempre es la más adecuada. Otra medida de tendencia central que se utiliza habitualmente es la **mediana**, que es el valor que aparece en el punto medio de una lista ordenada de valores. Al igual que con la media, R proporciona una función median(), que podemos aplicar a nuestros datos de salarios como se muestra en el siguiente ejemplo:

```
> median(c(36000, 44000, 56000))
[1] 44000
```

Por lo tanto, como el valor medio es 44 000, el ingreso medio es $44 000.

Si un conjunto de datos tiene un número par de valores, no hay un valor medio. En este caso, la mediana se calcula comúnmente como el promedio de los dos valores en el centro de la lista ordenada. Por ejemplo, la mediana de los valores 1, 2, 3 y 4 es 2.5.

A primera vista, parece que la mediana y la media son medidas muy similares.
Ciertamente, el valor medio de $45,333 y el valor medio de $44,000 no están muy alejados. ¿Por qué tener dos medidas de tendencia central? La razón se debe al hecho de que la media y la mediana se ven afectadas de manera diferente por los valores que caen en los extremos del rango. En particular, la media es muy sensible a los valores atípicos, o valores que son atípicamente altos o bajos en relación con la mayoría de los datos. Por lo tanto, debido a que la media es más sensible a los valores atípicos, es más probable que se desplace hacia arriba o hacia abajo por una pequeña cantidad de valores extremos.

Recuerda nuevamente los valores medianos informados en la salida summary() para el conjunto de datos de autos usados. Aunque la media y la mediana para el precio son bastante similares (difieren aproximadamente en un cinco por ciento), hay una diferencia mucho mayor entre la media y la mediana para el kilometraje. Para el kilometraje, la media de 44,261 es más del 20 por ciento mayor que la mediana de 36,385. Dado que la media es más sensible a los valores extremos que la mediana, el hecho de que la media sea mucho más alta que la

mediana podría llevarnos a sospechar que hay algunos autos usados en el conjunto de datos con valores de kilometraje extremadamente altos. Para investigar esto más a fondo, necesitaremos agregar estadísticas de resumen adicionales a nuestro análisis.

Medición de la dispersión: cuartiles y resumen de cinco números

La media y la mediana proporcionan formas de resumir rápidamente los valores, pero estas medidas de centro nos dicen poco acerca de si hay o no diversidad en las mediciones. Para medir la diversidad, necesitamos emplear otro tipo de estadísticas de resumen relacionadas con la **dispersión** de los datos, o cuán estrechamente o poco espaciados están los valores. Conocer la dispersión proporciona una idea de los máximos y mínimos de los datos, y si la mayoría de los valores son similares o diferentes a la media y la mediana.

El **resumen de cinco números** es un conjunto de cinco estadísticas que representan aproximadamente la dispersión de los valores de una característica. Las cinco estadísticas se incluyen en el resultado de la función summary(). Escritos en orden, son:

1. Mínimo (Min.)
2. Primer cuartil, o Q1 (1st Qu.)
3. Mediana, o Q2 (Median)
4. Tercer cuartil, o Q3 (3rd Qu.)
5. Máximo (Max.)

Como se esperaría, el mínimo y el máximo son los valores característicos más extremos, que indican los valores más pequeños y más grandes respectivamente. R proporciona las funciones min() y max() para calcularlos para un vector.

El intervalo entre el valor mínimo y el máximo se conoce como **rango**.
En R, la función range() devuelve tanto el valor mínimo como el máximo:

```
> range(usedcars$price)
[1]  3800  21992
```

La combinación de range() con la función de diferencia diff() te permite calcular la estadística de rango con una sola línea de código:

```
> diff(range(usedcars$price))
[1] 18192
```

El primer y tercer cuartil, Q1 y Q3, se refieren al valor por debajo y por encima del cual se encuentra una cuarta parte de los valores. Junto con la mediana (Q2), los **cuartiles** dividen un conjunto de datos en cuatro partes, cada una con la misma cantidad de valores.

Los cuartiles son un caso especial de un tipo de estadística llamada **cuantiles**, que son números que dividen los datos en cantidades de igual tamaño. Además de los cuartiles, los cuartiles que se usan comúnmente incluyen **terciles** (tres partes), **quintiles** (cinco partes), **deciles** (10 partes) y **percentiles** (100 partes).

Los percentiles se usan a menudo para describir la clasificación de un valor; por ejemplo, un estudiante cuya puntuación en una prueba se ubicó en el percentil 99 tuvo un desempeño mejor o igual al 99 por ciento de los demás examinados.

El 50 por ciento intermedio de los datos, que se encuentra entre el primer y el tercer cuartil, es de particular interés porque es una medida simple de dispersión. La diferencia entre Q1 y Q3 se conoce como **rango intercuartil** (RIC) y se puede calcular con la función IQR():

```
> IQR(usedcars$price)
[1] 3909.5
```

También podríamos haber calculado este valor a mano a partir del resultado summary() para la variable usedcars$price calculando 14904 – 10995 = 3909. La pequeña diferencia entre nuestro cálculo y el resultado IQR() se debe al hecho de que R redondea automáticamente el resultado summary().

La función quantile() proporciona una herramienta versátil para identificar cuantiles para un conjunto de valores. De forma predeterminada, la función quantile() devuelve el resumen de cinco números.
La aplicación de la función a la variable usedcars$price da como resultado las mismas estadísticas de resumen que antes:

```
> quantile(usedcars$price)
    0%    25%    50%    75%   100%
3800.0 10995.0 13591.5 14904.5 21992.0
```

Al calcular cuantiles, existen muchos métodos para manejar empates entre conjuntos de valores sin un único valor intermedio. La función quantile() te permite especificar entre nueve algoritmos de desempate diferentes especificando el parámetro de tipo. Si tu proyecto requiere un cuantil definido con precisión, es importante leer la documentación de la función utilizando el comando ?quantile.

Al especificar un parámetro probs adicional utilizando un vector que denota puntos de corte, podemos obtener cuantiles arbitrarios, como los percentiles 1 y 99:

```
> quantile(usedcars$price, probs = c(0.01, 0.99))
     1%         99%
 5428.69   20505.00
```

La función de secuencia seq() genera vectores de valores espaciados uniformemente. Esto facilita la obtención de otras porciones de datos, como los quintiles (cinco grupos) que se muestran en el siguiente comando:

```
> quantile(usedcars$price, seq(from = 0, to = 1, by = 0.20))
    0%      20%       40%       60%       80%      100%
 3800.0  10759.4   12993.8   13992.0   14999.0   21992.0
```

Conociendo el resumen de cinco números, podemos volver a examinar el resultado del resumen de autos usados, summary(). En la variable precio, el mínimo fue $3,800 y el máximo fue $21,992. Curiosamente, la diferencia entre el mínimo y Q1 es de aproximadamente $7,000, al igual que la diferencia entre Q3 y el máximo; sin embargo, la diferencia entre Q1 y la mediana y Q3 es de aproximadamente $2,000. Esto sugiere que el 25 por ciento inferior y superior de los valores están más dispersos que el 50 por ciento medio de los valores, que parecen estar agrupados más estrechamente alrededor del centro. También vemos una tendencia similar con la variable kilometraje. Como aprenderás más adelante en este documento, este patrón de dispersión es lo suficientemente común como para que se le haya llamado una distribución “normal” de datos.

La dispersión de la variable de kilometraje (mileage) también muestra otra propiedad interesante - la diferencia entre Q3 y el máximo es mucho mayor que la que existe entre el mínimo y Q1. En otras palabras, los valores mayores están mucho más dispersos que los valores menores.

Este hallazgo ayuda a explicar por qué el valor medio es mucho mayor que la mediana. Como la media es sensible a los valores extremos, se eleva, mientras que la mediana permanece relativamente en el mismo lugar. Esta es una propiedad importante, que se vuelve más evidente cuando los datos se presentan visualmente.

Visualización de variables numéricas: diagramas de caja

Visualizar variables numéricas puede ser útil para diagnosticar problemas de datos.
Una visualización común del resumen de cinco números es un diagrama de caja, también conocido como **diagrama de caja y bigotes**. El diagrama de caja muestra el centro y la

dispersión de una variable numérica en un formato que te permite obtener rápidamente una idea del rango y la asimetría de una variable o compararla con otras variables.

Echemos un vistazo a un diagrama de caja para los datos de kilometraje y precio de autos usados. Para obtener un diagrama de caja para una variable, utilizaremos la función boxplot(). También especificaremos un par de parámetros adicionales, main e ylab, para agregar un título a la figura y etiquetar el eje y (el eje vertical), respectivamente. Los comandos para crear los diagramas de caja de precio y kilometraje son:

```
> par(mfrow = c(1, 2)) # un renglón, dos columnas
> boxplot(usedcars$price, main = "Boxplot of Used Car Prices",
+   ylab = "Price ($)")
> boxplot(usedcars$mileage, main = "Boxplot of Used Car Mileage",
+   ylab = "Odometer (mi.)")
```

R producirá las siguientes figuras:

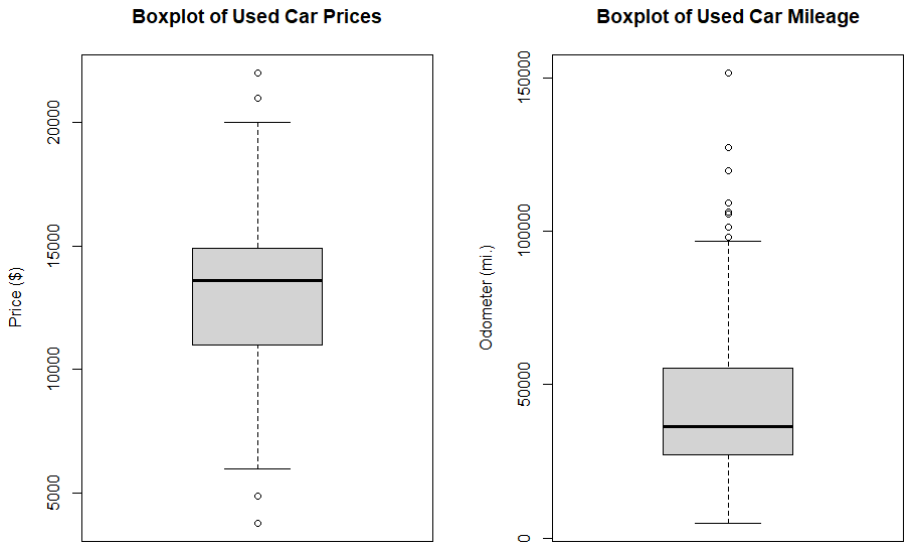

Figura 2.1: Diagramas de caja de datos de kilometraje y precio de autos usados.

Un diagrama de caja representa el resumen de cinco números mediante líneas horizontales y puntos.

Las líneas horizontales que forman el cuadro en el medio de cada figura representan Q1, Q2 (la mediana) y Q3 cuando se lee el gráfico de abajo hacia arriba. La mediana se denota por la línea oscura, que se alinea con $13,592 en el eje vertical para el precio y 36,385 mi. en el eje vertical para el kilometraje.

En diagramas de caja simples, como los del diagrama anterior, el ancho de la caja es arbitrario y no ilustra ninguna característica de los datos. Para análisis más sofisticados, es posible

utilizar la forma y el tamaño de las cajas para facilitar las comparaciones de los datos entre varios grupos. Para obtener más información sobre dichas características, comienza examinando las opciones notch y varwidth en la documentación de boxplot() de R escribiendo el comando ?boxplot.

Los valores mínimos y máximos se pueden ilustrar utilizando los bigotes que se extienden por debajo y por encima de la caja; Sin embargo, una convención ampliamente utilizada solo permite que los bigotes se extiendan hasta un mínimo o máximo de 1.5 veces el RIQ por debajo de Q1 o por encima de Q3.

Cualquier valor que caiga más allá de este umbral se considera atípico y se denota como círculos o puntos. Por ejemplo, recuerda que el RIQ para la variable precio fue 3,909 con Q1 de 10,995 y Q3 de 14,904. Por lo tanto, un valor atípico es cualquier valor que sea menor que 10,995 – 1.5 * 3,909 = 5,131.5 o mayor que 14,904 + 1.5 * 3,909 = 20,767.5.

El diagrama de caja de precio muestra dos valores atípicos en los extremos superior e inferior. En el diagrama de caja de kilometraje, no hay valores atípicos en el extremo inferior y, por lo tanto, el bigote inferior se extiende hasta el valor mínimo de 4,867. En el extremo superior, vemos varios valores atípicos más allá de la marca de 100,000 millas.

Estos valores atípicos son responsables de nuestro hallazgo anterior, que señaló que el valor medio era mucho mayor que la mediana.

Visualización de variables numéricas: histogramas

Un **histograma** es otra forma de visualizar la dispersión de una variable numérica. Es similar a un diagrama de caja en el sentido de que divide los valores de la variable en una cantidad predefinida de porciones o compartimentos que actúan como contenedores de valores. Sin embargo, sus similitudes terminan allí.
Mientras que un diagrama de caja crea cuatro porciones que contienen la misma cantidad de valores pero que varían en rango, un histograma utiliza una cantidad mayor de porciones de rango idéntico y permite que los compartimentos contengan diferentes cantidades de valores.

Podemos crear un histograma para los datos de kilometraje (no olvidar que se usan millas) y precio de autos usados utilizando la función hist(). Como hicimos con el diagrama de caja, especificaremos un título para la figura utilizando el parámetro principal y etiquetaremos el eje x con el parámetro xlab. Los comandos para crear los histogramas son:

```
> par(mfrow = c(1, 2))
>  hist(usedcars$price, main = "Histogram of Used Car Prices",
+   xlab = "Price ($)")
```

```
> hist(usedcars$mileage, main = "Histogram of Used Car Mileage",
+   xlab = "Odometer (mi.)")
```

Esto produce los siguientes diagramas:

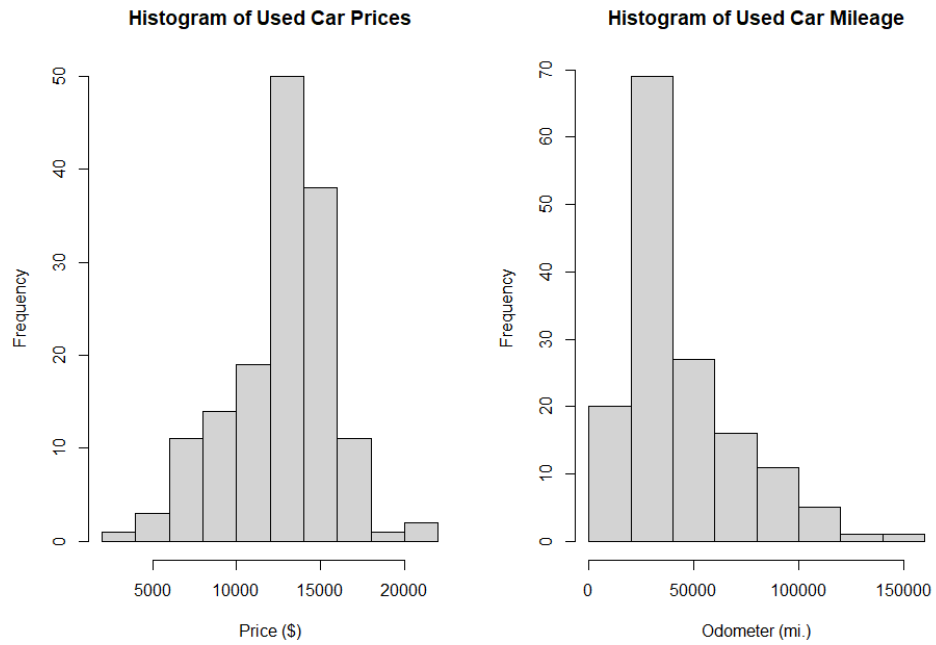

Figura 2.2: Histogramas de datos de millas y precio de autos usados.

El histograma está compuesto por una serie de barras cuyas alturas indican el recuento o la **frecuencia** de los valores que caen dentro de cada uno de los intervalos de igual ancho que dividen los valores. Las líneas verticales que separan las barras, como se indica en el eje horizontal, indican los puntos de inicio y fin del rango de valores que caen dentro del intervalo.

Es posible que hayas notado que los histogramas anteriores tienen diferentes cantidades de intervalos. Esto se debe a que la función hist() intenta identificar la cantidad óptima de intervalos para el rango de la variable. Si deseas anular este valor predeterminado, utiliza el parámetro breaks. Al proporcionar un número entero como breaks = 10 se crean exactamente 10 contenedores de igual ancho, mientras que al proporcionar un vector como c(5000, 10000, 15000, 20000) se crean contenedores que se dividen en los valores especificados.

En el histograma de precios, cada una de las 10 barras abarca un intervalo de 2,000 dólares, comenzando en 2,000 y terminando en 22,000 dólares. La barra más alta en el centro de la figura cubre el rango de 12,000 a 14,000 dólares y tiene una frecuencia de 50. Como sabemos que nuestros datos incluyen 150 coches, sabemos que un tercio de todos los coches tienen un precio de entre 12,000 y 14,000 dólares. Casi 90 coches (más de la mitad) tienen un precio de entre 12,000 y 16,000 dólares.

El histograma de millas incluye ocho barras que representan intervalos de 20,000 millas cada uno, comenzando en 0 y terminando en 160,000 millas. A diferencia del histograma de precios, la barra más alta no está en el centro de los datos, sino en el lado izquierdo del diagrama. Los 70 autos que contiene este contenedor tienen lecturas de odómetro de entre 20,000 y 40,000 millas.

También puede notar que la forma de los dos histogramas es algo diferente. Parece que los precios de los autos usados tienden a dividirse de manera uniforme en ambos lados del medio, mientras que las millas de los autos se extienden más hacia la derecha.

Esta característica se conoce como **sesgo (inclinación!)**, o más específicamente sesgo a la derecha, porque los valores en el extremo superior (lado derecho) están mucho más dispersos que los valores en el extremo inferior (lado izquierdo). Como se muestra en el siguiente diagrama, los histogramas de datos sesgados se ven estirados en uno de los lados:

Figura 2.3: Tres patrones de sesgo visualizados con histogramas idealizados.

La capacidad de diagnosticar rápidamente dichos patrones en nuestros datos es una de las fortalezas del histograma como herramienta de exploración de datos. Esto se volverá aún más importante a medida que comencemos a examinar otros patrones de dispersión en datos numéricos.

Comprensión de los datos numéricos: distribuciones uniformes y normales

Los histogramas, diagramas de caja y estadísticas que describen el centro y la dispersión proporcionan formas de examinar la distribución de los valores de una variable. La distribución de una variable describe la probabilidad de que un valor se encuentre dentro de varios rangos.

Si todos los valores tienen la misma probabilidad de ocurrir (por ejemplo, en un conjunto de datos que registra los valores que se lanzan en un dado de seis caras), se dice que la distribución es uniforme.

Una distribución uniforme es fácil de detectar con un histograma porque las barras tienen aproximadamente la misma altura. El histograma puede verse como el siguiente diagrama:

Figura 2.4: Una distribución uniforme visualizada con un histograma idealizado.

Es importante tener en cuenta que no todos los eventos aleatorios son uniformes. Por ejemplo, al lanzar un dado de seis caras con truco y ponderado, algunos números saldrán con más frecuencia que otros. Si bien cada lanzamiento del dado da como resultado un número seleccionado al azar, no tienen la misma probabilidad.

Tomemos, por ejemplo, los datos de las millas y precio de los autos usados. Esto claramente no es uniforme, ya que algunos valores parecen tener una probabilidad mucho mayor de ocurrir que otros. De hecho, en el histograma de precios, parece que los valores tienen menos probabilidad de ocurrir cuanto más se alejan de ambos lados de la barra central, lo que da como resultado una distribución de datos en forma de campana. Esta característica es tan común en los datos del mundo real que es el sello distintivo de la denominada distribución normal. La curva estereotipada en forma de campana de la distribución normal se muestra en el siguiente diagrama:

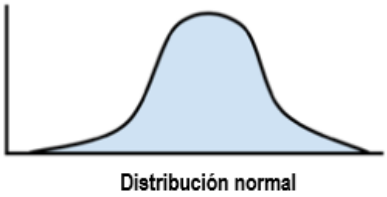

Figura 2.5: Una distribución normal visualizada con un histograma idealizado.

Aunque existen numerosos tipos de distribuciones no normales, muchos fenómenos del mundo real generan datos que pueden describirse mediante la distribución normal.
Por lo tanto, las propiedades de la distribución normal se han estudiado en gran detalle.

Medición de la dispersión: varianza y desviación típica

Las distribuciones nos permiten caracterizar una gran cantidad de valores utilizando una menor cantidad de parámetros. La distribución normal, que describe muchos tipos de datos del mundo real, se puede definir con solo dos: centro y dispersión. El centro de la distribución normal se define por su valor medio, que hemos utilizado antes. La dispersión se mide mediante una estadística llamada **desviación típica** (estándar).

Para calcular la desviación típica, primero debemos obtener la varianza, que se define como el promedio de las diferencias al cuadrado entre cada valor y el valor medio. En notación matemática, la varianza de un conjunto de *n* valores en un conjunto llamado *x* se define mediante la siguiente fórmula. La letra griega mu (similar en apariencia a una m o u) denota la media de los valores, y la varianza en sí se denota por la letra griega sigma al cuadrado (similar a una b girada de lado):

$$\text{Var}(X) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

La desviación estándar es la raíz cuadrada de la varianza y se denota por sigma como se muestra en la siguiente fórmula:

$$\text{StdDev}(X) = \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

En R, las funciones var() y sd() se pueden utilizar para obtener la varianza y la desviación estándar. Por ejemplo, al calcular la varianza y la desviación estándar en los vectores de precio y millas, encontramos:

```
>  var(usedcars$price)
[1] 9749892
> sd(usedcars$price)
[1] 3122.482
> var(usedcars$mileage)
[1] 728033954
> sd(usedcars$mileage)
[1] 26982.1
```

Al interpretar la varianza, los números más grandes indican que los datos están más dispersos alrededor de la media. La desviación estándar indica, en promedio, cuánto difiere cada valor de la media.

Si calculas estas estadísticas a mano utilizando las fórmulas de los diagramas anteriores, obtendrás un resultado ligeramente diferente al de las funciones integradas de R. Esto se debe a que las fórmulas anteriores utilizan la varianza de la población (que se divide por *n*), mientras que R utiliza la varianza de la muestra (que se divide por *n* - 1). Excepto en el caso de conjuntos de datos muy pequeños, la distinción es menor.

La desviación estándar se puede utilizar para estimar rápidamente cuán extremo es un valor dado bajo el supuesto de que proviene de una distribución normal. La regla **68-95-99.7** establece que el 68 por ciento de los valores en una distribución normal se encuentran dentro de una desviación estándar de la media, mientras que el 95 por ciento y el 99.7 por ciento de los valores se encuentran dentro de dos y tres desviaciones estándar, respectivamente. Esto se ilustra en el siguiente diagrama:

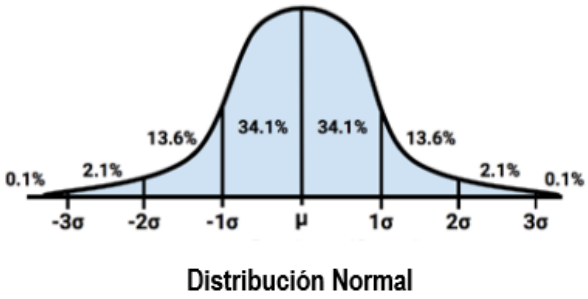

Figura 2.6: El porcentaje de valores dentro de una, dos y tres desviaciones estándar de la media de una distribución normal.

Al aplicar esta información a los datos de autos usados, sabemos que la media y la desviación estándar del precio fueron $12,962 y $3,122 respectivamente. Por lo tanto, al suponer que los precios se distribuyen normalmente, aproximadamente el 68 por ciento de los automóviles en nuestros datos se anunciaron a precios entre $12,962 - $3,122 = $9,840 y $12,962 + $3,122 = $16,804.

Aunque, estrictamente hablando, la regla 68–95–99.7 solo se aplica a distribuciones normales, el principio básico se aplica a cualquier dato; los valores que se alejan más de tres desviaciones estándar de la media son eventos extremadamente raros.

Exploración de variables categóricas

Si recuerdas, el conjunto de datos de automóviles usados contiene tres variables categóricas: modelo, color y transmisión. R las almacenó como vectores de caracteres (chr) en lugar de tipo de factor porque usamos el parámetro stringsAsFactors = FALSE al cargar los datos. Además, aunque la variable del año se almacena como un vector numérico (int), cada año se puede imaginar como una categoría aplicada a múltiples automóviles.
Por lo tanto, también podemos considerar tratarla como categórica.

A diferencia de los datos numéricos, los datos categóricos suelen examinarse mediante tablas en lugar de estadísticas resumidas. Una tabla que presenta una única variable categórica se conoce como **tabla unidireccional** (one-way-table). La función table() se puede utilizar para generar tablas unidireccionales para los datos de automóviles usados:

```
> table(usedcars$year)

2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
   3    1    1    1    3    2    6   11   14   42   49   16    1
> table(usedcars$model)

 SE SEL SES
 78  23  49
> table(usedcars$color)

 Black   Blue   Gold   Gray  Green    Red Silver  White Yellow
    35     17      1     16      5     25     32     16      3
```

La salida de table() enumera las categorías de la variable nominal y un recuento de la cantidad de valores que caen en cada categoría. Como sabemos que hay 150 autos usados en el conjunto de datos, podemos determinar que aproximadamente un tercio de todos los autos se fabricaron en 2010, dado que 49 / 150 = 0.327.

R también puede realizar el cálculo de las proporciones de la tabla directamente, utilizando el comando prop.table() en una tabla producida por la función table():

```
> model_table <- table(usedcars$model)
> prop.table(model_table)

      SE          SEL          SES
0.5200000   0.1533333   0.3266667
```

Los resultados de prop.table() se pueden combinar con otras funciones de R para transformar la salida. Supongamos que nos gustaría mostrar los resultados en porcentajes con un solo decimal. Podemos hacer esto multiplicando las proporciones por 100 y luego utilizando la función round() mientras especificamos digits = 1, como se muestra en el siguiente ejemplo:

```
> color_table <- table(usedcars$color)
> color_pct <- prop.table(color_table) * 100
> round(color_pct, digits = 1)

 Black   Blue   Gold   Gray  Green    Red Silver  White Yellow
  23.3   11.3    0.7   10.7    3.3   16.7   21.3   10.7    2.0
```

Aunque esto incluye la misma información que la salida predeterminada de prop.table(), los cambios facilitan la lectura. Los resultados muestran que el negro es el color más común, con casi una cuarta parte (23.3 por ciento) de todos los coches anunciados. El plateado es el segundo con un 21.3 por ciento, y el rojo el tercero con un 16.7 por ciento.

Medición de la tendencia central: la moda

En la terminología estadística, la **moda** de una característica es el valor que aparece con mayor frecuencia.
Al igual que la media y la mediana, la moda es otra medida de tendencia central. Se utiliza normalmente para datos categóricos, ya que la media y la mediana no están definidas para las variables nominales.

Por ejemplo, en los datos de coches usados, la moda de la variable del año es 2010, mientras que las modas de las variables de modelo y color son SE y Black, respectivamente. Una variable puede tener más de una moda; una variable con una sola moda es **unimodal**, mientras que una variable con dos modas es **bimodal**. Los datos con múltiples modas se denominan más generalmente **multimodales**.

Aunque podrías sospechar que podrías utilizar la función mode(), R la utiliza para obtener el tipo de variable (como en numérica, lista, etc.) en lugar de la moda estadística. En cambio, para encontrar la moda estadística, simplemente observa la salida de table() para la categoría con la mayor cantidad de valores.

La moda o las modas se utilizan en un sentido cualitativo para comprender los valores importantes. Aun así, sería peligroso poner demasiado énfasis en la moda, ya que el valor más común no es necesariamente mayoritario. Por ejemplo, aunque el negro era el color de automóvil más común, solo representaba aproximadamente una cuarta parte de todos los automóviles publicitados.

Es mejor pensar en las modas en relación con las otras categorías. ¿Hay una categoría que domina a todas las demás o hay varias? Pensar en las modas de esta manera puede ayudar a generar hipótesis comprobables al plantear preguntas sobre qué hace que ciertos valores sean más comunes que otros. Si el negro y el plateado son colores comunes de automóviles usados, podríamos creer que los datos representan automóviles de lujo, que tienden a venderse en colores más conservadores. Alternativamente, estos colores podrían indicar automóviles económicos, que se venden con menos opciones de color. Tendremos estas preguntas en mente a medida que continuamos examinando estos datos.

Pensar en las modas como valores comunes nos permite aplicar el concepto de moda estadística a los datos numéricos. En sentido estricto, sería poco probable que exista una moda para una variable continua, ya que no es probable que se repitan dos valores. Sin embargo, si pensamos en las modas como las barras más altas de un histograma, podemos analizar las modas de variables como el precio y las millas. Puede resultar útil considerar la moda al explorar datos numéricos, en particular para examinar si los datos son multimodales o no.

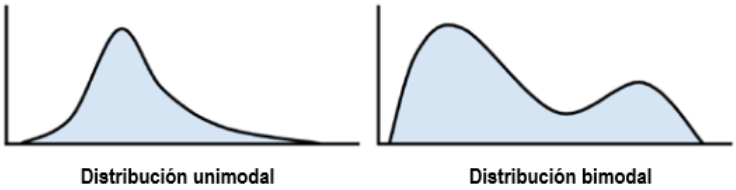

Figura 2.7: Distribuciones hipotéticas de datos numéricos con una y dos modas.

Exploración de las relaciones entre variables

Hasta ahora, hemos examinado las variables de a una por vez, calculando solo estadísticas univariadas. Durante nuestra investigación, planteamos preguntas que no pudimos responder antes:

- ¿Los datos de price y mileage implican que estamos examinando solo autos de clase económica o hay autos de lujo con alto kilometraje?
- ¿Las relaciones entre el modelo y el color brindan información sobre los tipos de autos que estamos examinando?

Este tipo de preguntas se pueden abordar analizando las relaciones **bivariadas**, que consideran la relación entre dos variables. Las relaciones de más de dos variables se denominan relaciones **multivariadas**. Comencemos con el caso bivariado.

Visualización de relaciones: diagramas de dispersión (scatterplots)

Un **diagrama de dispersión** es un diagrama que visualiza una relación bivariada entre características numéricas. Es una figura bidimensional en la que se dibujan puntos en un plano de coordenadas utilizando los valores de una característica para proporcionar las coordenadas *x* horizontales y los valores de otra característica para proporcionar las coordenadas *y* verticales. Los patrones en la colocación de los puntos revelan asociaciones subyacentes entre las dos características.

Para responder a nuestra pregunta sobre la relación entre el precio y las millas, examinaremos un diagrama de dispersión. Utilizaremos la función plot(), junto con los parámetros main, xlab e ylab utilizados anteriormente para etiquetar el diagrama.

Para utilizar plot(), necesitamos especificar los vectores *x* e *y* que contienen los valores utilizados para posicionar los puntos en la figura. Aunque las conclusiones serían las mismas independientemente de la variable utilizada para proporcionar las coordenadas *x* e *y*, la convención dicta que la variable *y* es la que se supone que depende de la otra (y, por lo tanto, se la conoce como la **variable dependiente**). Dado que un vendedor no puede modificar la

lectura del odómetro de un automóvil, es poco probable que las millas dependan del precio del automóvil.
En cambio, nuestra hipótesis es que el precio de un automóvil depende de las millas del odómetro.
Por lo tanto, seleccionaremos el precio como la variable *y* dependiente.
El comando completo para crear nuestro diagrama de dispersión es:

```
> par(mfrow = c(1, 1))
> plot(x = usedcars$mileage, y = usedcars$price,
+   main = "Scatterplot of Price vs. Mileage",
+   xlab = "Used Car Odometer (mi.)",
+   ylab = "Used Car Price ($)")
```

Cuando se usa la impresión de varios diagramas en una misma figura, es importante resetear a una sola cuando sea necesario, de lo contrario las figuras saldrán dispuestas como una matriz (se usa el comando par(mfrow = c(1, 1)) para restaurar a una sola figura).

Esto da como resultado el siguiente diagrama de dispersión:

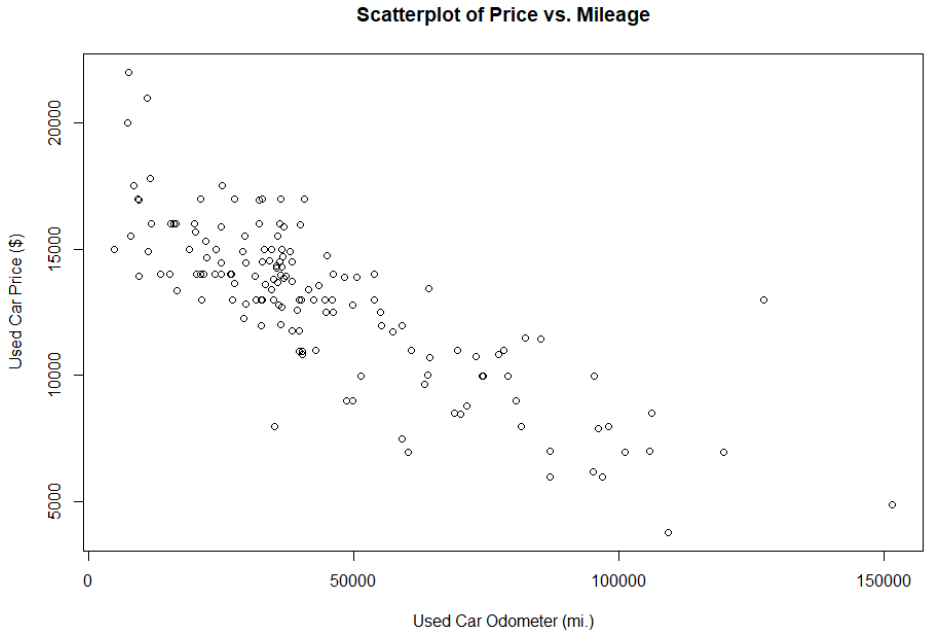

Figura 2.8: Relación entre el precio de un automóvil usado y las millas.

Usando el diagrama de dispersión, notamos una clara relación entre el precio de un automóvil usado y la lectura del odómetro. Para leer el gráfico, examina cómo cambian los valores de la variable del eje *y* a medida que aumentan los valores del eje *x*. En este caso, los precios de los automóviles tienden a ser más bajos a medida que aumentan las millas. Si alguna vez has vendido o comprado un automóvil usado, esto no es una idea profunda.

Quizás un hallazgo más interesante es el hecho de que hay muy pocos automóviles que tengan un precio alto y un alto número de millas, aparte de un caso atípico solitario de aproximadamente 125,000 millas y $14,000. La ausencia de más puntos como este proporciona evidencia para respaldar una conclusión de que es poco probable que nuestro conjunto de datos incluya automóviles de lujo con alto kilometraje. Todos los coches más caros de los datos, en particular los que superan los 17,500 dólares, parecen tener unas millas extraordinariamente bajas, lo que implica que podríamos estar viendo un único tipo de coche que se vende a un precio de alrededor de 20,000 dólares cuando es nuevo.

La relación que hemos observado entre los precios de los automóviles y las millas se conoce como asociación negativa porque forma un patrón de puntos en una línea que se inclina hacia abajo.

Una asociación positiva parecería formar una línea que se inclina hacia arriba. Una línea plana, o una dispersión aparentemente aleatoria de puntos, es evidencia de que las dos variables no están asociadas en absoluto. La fuerza de una asociación lineal entre dos variables se mide mediante una estadística conocida como correlación. Las correlaciones se analizan posteriormente.

Ten en cuenta que no todas las asociaciones forman líneas rectas. A veces, los puntos forman una forma de U o de V, mientras que a veces el patrón parece ser más débil o más fuerte para valores crecientes de la variable *x* o *y*. Estos patrones implican que la relación entre las dos variables no es lineal.

Examen de relaciones: tabulaciones cruzadas de dos vías

Para examinar una relación entre dos variables nominales, se utiliza una **tabulación cruzada de dos vías** (two-way cross-tabulation) (también conocida como **crosstab** o **tabla de contingencia**). Una tabulación cruzada es similar a un diagrama de dispersión en el sentido de que permite examinar cómo varían los valores de una variable en función de los valores de otra. El formato es una tabla en la que las filas son los niveles de una variable, mientras que las columnas son los niveles de otra.
Los recuentos en cada una de las celdas de la tabla indican la cantidad de valores que caen en la combinación particular de fila y columna.

Para responder a nuestra pregunta anterior sobre si existe una relación entre el modelo y el color, examinaremos una tabla de contingencia. Existen varias funciones para producir tablas de dos vías en R, incluida table(), que usamos para tablas de una vía. La función CrossTable() en el paquete gmodels de Gregory R. Warnes es quizás la más fácil de usar, ya que presenta los porcentajes de fila, columna y margen en una sola tabla, lo que nos ahorra el problema de calcularlos nosotros mismos. Para instalar el paquete gmodels, escribe:

```
> install.packages("gmodels")
> library(gmodels)
```

Después de instalar el paquete, escribe library(gmodels) para cargarlo. Deberás hacer esto durante cada sesión de R en la que planees usar la función CrossTable().

Antes de continuar con nuestro análisis, simplifiquemos nuestro proyecto reduciendo la cantidad de niveles en la variable de color. Esta variable tiene nueve niveles, pero en realidad no necesitamos tantos detalles. Lo que realmente nos interesa es si el color del automóvil es conservador o no. Con este fin, dividiremos los nueve colores en dos grupos: el primer grupo incluirá los colores conservadores negro, gris, plateado y blanco; el segundo grupo incluirá azul, dorado, verde, rojo y amarillo. Crearemos una variable indicadora binaria (a menudo llamada **variable ficticia** (dummy variable)), que indicará si el color del automóvil es conservador o no según nuestra definición. Su valor será 1 si es verdadero y 0 en caso contrario:

```
> usedcars$conservative <-
+   usedcars$color %in% c("Black", "Gray", "Silver", "White")
```

Es posible que hayas notado un nuevo comando aquí: el operador %in% devuelve VERDADERO o FALSO para cada valor en el vector del lado izquierdo del operador, indicando si el valor se encuentra en el vector del lado derecho. En términos simples, puedes traducir esta línea como "¿el color del auto usado está en el conjunto de negro, gris, plateado y blanco?"

Al examinar la salida de table() para nuestra variable recién creada, vemos que aproximadamente dos tercios de los autos tienen colores conservadores mientras que un tercio no:

```
> table(usedcars$conservative)

FALSE  TRUE
   51    99
```

Ahora, veamos una tabulación cruzada para ver cómo varía la proporción de autos con colores conservadores según el modelo. Dado que asumimos que el modelo del auto determina la elección del color, trataremos el indicador de color conservador como la variable dependiente (*y*). Por lo tanto, el comando CrossTable() es:

```
> CrossTable(x = usedcars$model, y = usedcars$conservative)
```

Esto da como resultado la siguiente tabla:

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  150


                | usedcars$conservative
usedcars$model |     FALSE |      TRUE | Row Total |
---------------|-----------|-----------|-----------|
            SE |        27 |        51 |        78 |
               |     0.009 |     0.004 |           |
               |     0.346 |     0.654 |     0.520 |
               |     0.529 |     0.515 |           |
               |     0.180 |     0.340 |           |
---------------|-----------|-----------|-----------|
           SEL |         7 |        16 |        23 |
               |     0.086 |     0.044 |           |
               |     0.304 |     0.696 |     0.153 |
               |     0.137 |     0.162 |           |
               |     0.047 |     0.107 |           |
---------------|-----------|-----------|-----------|
           SES |        17 |        32 |        49 |
               |     0.007 |     0.004 |           |
               |     0.347 |     0.653 |     0.327 |
               |     0.333 |     0.323 |           |
               |     0.113 |     0.213 |           |
---------------|-----------|-----------|-----------|
  Column Total |        51 |        99 |       150 |
               |     0.340 |     0.660 |           |
---------------|-----------|-----------|-----------|
```

Hay una gran cantidad de datos en la salida de CrossTable(). La leyenda en la parte superior (etiquetada Contenido de la celda, Cell Contents) indica cómo interpretar cada valor. Las filas de la tabla indican los tres modelos de autos usados: SE, SEL y SES (más una fila adicional para el total de todos los modelos). Las columnas indican si el color del auto es conservador o no (más una columna que suma ambos tipos de color).

El primer valor en cada celda indica la cantidad de autos con esa combinación de modelo y color. Las proporciones indican la contribución de cada celda a la estadística de chi-cuadrada, el total de filas, el total de columnas y el total general de la tabla.

Lo que más nos interesa es la proporción de autos conservadores para cada modelo. Las proporciones de filas nos indican que 0.654 (65 por ciento) de los autos SE tienen colores conservadores, en comparación con 0.696 (70 por ciento) de los autos SEL y 0.653 (65 por ciento) de los SES. Estas diferencias son relativamente pequeñas, lo que sugiere que no hay diferencias sustanciales en los tipos de colores elegidos para cada modelo de auto.

Los valores de chi-cuadrada se refieren a la contribución de la celda en la **prueba de chi-cuadrada de Pearson para la independencia** entre dos variables. Esta prueba mide la probabilidad de que la diferencia en los recuentos de celdas en la tabla se deba únicamente al azar. Si la probabilidad es muy baja, proporciona una evidencia sólida de que las dos variables están asociadas.

Puedes obtener los resultados de la prueba de chi-cuadrada agregando un parámetro adicional que especifique chisq = TRUE al llamar a la función CrossTable(). En este caso, la probabilidad es de alrededor del 93 por ciento, lo que sugiere que es muy probable que las variaciones en el recuento de celdas se deban solo al azar y no a una verdadera asociación entre el modelo y el color.

Resumen

En este documento, aprendimos los conceptos básicos de la gestión de datos en R. Comenzamos analizando en profundidad las estructuras utilizadas para almacenar varios tipos de datos.

La estructura de datos fundamental de R es el vector, que se extiende y se combina en tipos de datos más complejos, como listas y frames de datos. El frame de datos es una estructura de datos de R que corresponde a la noción de un conjunto de datos que tiene características y ejemplos. R proporciona funciones para leer y escribir frames de datos en archivos de datos tabulares similares a hojas de cálculo.

Luego, exploramos un conjunto de datos del mundo real que contiene precios de automóviles usados. Examinamos variables numéricas utilizando estadísticas de resumen comunes de centro y dispersión, y visualizamos las relaciones entre los precios y las lecturas del odómetro con un diagrama de dispersión.

A continuación, examinamos las variables nominales utilizando tablas. Al examinar los datos de los autos usados, seguimos un proceso exploratorio que se puede utilizar para comprender cualquier conjunto de datos.
Estas habilidades serán necesarias para los demás proyectos de este curso.

Ahora que hemos dedicado algo de tiempo a comprender los conceptos básicos de la gestión de datos con R, estás listo para comenzar a usar el aprendizaje automático para resolver problemas del mundo real. En los próximos temas, abordaremos tareas de clasificación utilizando varios métodos.