Informe Dinámico

Daniela Cuesta - Paola Peralta Flores

Capítulo 1

Introducción al Aprendizaje Automático

El aprendizaje automático también conocido como inteligencia artificial pretende ayudar a dar sentido a los grupos masivos de datos del mundo.

Los orígenes del aprendizaje automático

Nuestros sensores, los ojos, oidos, nariz, lengua y los nervios receptan datos sin procesar que el cerebro los traduce en imágenes, sonidos, olores, sabores y texturas. Actualmente, todo el procesamiento de datos, están cada vez más automatizados y registrados sistemáticamente en bases de datos computarizadas.

Esta automatización de registro de datos se ha visto beneficiada por el desarrollo de sensores electrónicos. Pese, a que estos sensores procesan los datos de manera muy diferente a como lo haría un ser humano los datos siguen siendo objetivos.

Nota: Aunque un sensor no tiene un componente subjetivo en sus observaciones, no necesariamente reporta la verdad.

Entre bases de datos y sensores se registran muchos aspectos de nuestra vida.Esta cantidad de datos ha llevado a pensar que hemos entrado en una era de "Big Data". Los conjuntos de datos más grandes e interesantes son facilmente accesibles con una búsqueda en la web. Incluso, si se logra dar sentido a todo de manera sistemática, esta información serviría a la toma de decisiones.

El campo de estudio interesado en el desarrollo de algoritmos informáticos para transformar datos en acciones inteligentes se conoce como **aprendizaje automático**. Este campo se originó en un entorno donde los datos disponibles, los métodos estadísticos y el poder de cómputo evolucionaron rápida y simultáneamente.

La minería de datos, se ocupa de la generación de información novedosa a partir de grandes bases de datos. A diferencia del aprendizaje automático, que tiende a centrarse en realizar una tarea conocida.

Usos y Abusos del aprendizaje automático

El aprendizaje automático se interesa principalmente por dar sentido a datos complejos, por ejemplo:

- Predecir los resultados de las elecciones.
- Identificar y filtrar mensajes de spam del correo electrónico.
- Automatizar los semáforos.
- Realizar estimaciones de tormentas y catástrofes naturales.
- Crear aviones autopilotados y coches de conducción automática

En todos los casos mencionados se aplica un algoritmo de aprendizaje automático, el cual toma datos e identifica patrones para que se realice lo requerido.

Nota: Actualmente se utilizan métodos de aprendizaje automático para la publicidad. Los sitios web lo hacen para ofrecer publicidad basada en el historial de navegación.

Consideraciones éticas

Debido a la relativa juventud del aprendizaje automático como disciplina y la velocidad a la que avanza, las cuestiones legales y las normas sociales asociadas están en constante cambio.

Ciertas jurisdicciones pueden impedirle usar datos raciales, étnicos, religiosos u otras clases protegidas por motivos comerciales: los algoritmos de aprendizaje automático pueden aprender esta información de forma independiente sin darse cuenta.

Aparte de las consecuencias legales, el uso inapropiado de ciertos datos privados puede perjudicar sus resultados. Recientemente, varias aplicaciones web han experimentado transferencias masivas de usuarios que se sintieron explotados cuando los términos de los acuerdos de servicio cambiaron y sus datos se usaron para propósitos más allá de lo acordado originalmente.

¿Cómo aprenden las máquinas?

Segun el informático Tom M. Mitchell, la definición del aprendizaje automático, es la transformación de los datos en acciones, lo cual nos proporciona una visión de las distinciones entre los algoritmos de aprendizaje automático.

El proceso básico de aprendizaje es similar y puede dividirse en los siguientes componentes:

- **Entrada de datos:** Utiliza la observación, el almacenamiento en la memoria.
- **Abstracción:** Consiste en traducir los datos en representaciones más amplias.
- **Generalización:** Utiliza los datos abstraídos para formar una base para la acción R.

Un ejemplo claro de esto es, cuando se tiene un examen complicado es complicado memorizar las respuestas a todas las preguntas, es por eso que lo que se tiene que dedicar tiempo a manejar un conjunto más reducido de ideas claves.

Las estrategias de aprendizaje más utilizadas, como la creación de un esquema o mapa conceptual, son similares a la forma en que una máquina realiza su trabajo.

Estas herramientas permiten definir las relación que existe entre la información proporcionada, además de qué representan las ideas dificiles sin necesidad de memorizarlas palabra por palabra, es decir, que se puede considerar estas herramientas una forma mas avanzada de aprendizaje.

Las etapas de abstracción y generalización son inseparables, ya que los seres humanos recordamos, deducimos, inducimos e intuimos, pero para un ordenador, estos procesos deben hacerse explícitos.

Abstracción y representación del conocimiento

La representación de datos de entrada sin procesar en un formato estructurado es ideal para un algoritmo de aprendizaje. En un inicio, los datos son binarios. La asignación de un significado a los datos se da en el proceso de abstracción.

La formación de estructuras lógicas ayuda a convertir la información sensorial sin procesar en una percepción significativa.

Durante el proceso de representación del conocimiento, la computadora resume las entradas sin procesar en un modelo, una descripción explícita de los patrones estructurados entre los datos. Hay muchos tipos de modelos, como:

- Ecuaciones
- Diagramas
- Reglas lógicas if/else

- Agrupaciones de datos (clústeres)

La elección del modelo no se deja a decisión de la máquina. El modelo está dictado por la tarea de aprendizaje y el tipo de datos que se analizan.

El proceso de ajustar un modelo particular a un conjunto de datos se conoce como entrenamiento, que sugiere que el maestro humano impone el modelo de aprendizaje automático al estudiante máquina. Mientras el aprendizaje implica una especie de razonamiento inductivo de abajo hacia arriba.

Una vez que el modelo ha sido entrenado, los datos se han transformado en una forma abstracta que resume la información original.

Generalización

Describe el proceso de convertir el conocimiento abstracto en forma que pueda utilizarse para la acción, implica la reducción a un conjunto manejable, lo que sucede es que los algoritmos de aprendizaje automático usan atajos que dividen el conjunto de conceptos, con esto, el algoritmo empleará heurísticas, o conjeturas.

La heurística empleada por los algoritmos de aprendizaje automático también suelen ser erróneas. Si las conclusiones son sistemáticamente imprecisas, se dice que el algoritmo tiene un sesgo.

Los seres humanos utilizan la heurística para generalizar la experiencia a nuevos escenarios, esto sucede cuando se utiliza el instinto para tomar una decisión.

Evaluar el éxito del aprendizaje

El paso final en el proceso de generalización es determinar el éxito del modelo a pesar de sus sesgos.

Una vez que un modelo ha sido entrenado en un conjunto de datos inicial, se prueba en un nuevo conjunto de datos y se juzga en qué medida su caracterización de los datos de entrenamiento se generaliza a los nuevos datos.

Los modelos no generalizan perfectamente por el ruido o variaciones en los datos. Los datos ruidosos son causados por:

- Error de medición debido a sensores imprecisos.
- Problemas con el informe de datos, respuestas aleatorias.
- Errores causados cuando los datos se registran incorrectamente.

Tratar de modelar el ruido en los datos es la base de un problema llamado sobreajuste, es decir, un modelo funciona bien durante el entrenamiento pero mal durante las pruebas. Las soluciones a este problema son específicas para enfoques particulares de aprendizaje automático.

Pasos para aplicar el aprendizaje automático a sus datos

El aprendizaje automático se divide en los siguientes pasos:

1. **Recopilación de datos:** Si los datos están escritos en papel, registrados en archivos tendrán que estar formato electrónico adecuado para el análisis.

2. Explorar y preparar los datos: La calidad de cualquier proyecto de aprendizaje se basa en la calidad de los datos que utiliza, esto se denomina exploración de datos.

3. Entrenamiento de un modelo a partir de los datos: Cuando los datos se han preparado para el análisis, se tiene que que tener una idea de lo que se va a esperar aprender de los datos. La tarea específica de aprendizaje automático dara a conocer la selección de un algoritmo, el cual representará los datos en forma de modelo.

4. Evaluar el rendimiento del modelo: Dado que cada modelo de aprendizaje automático resulta en una solución del problema de aprendizaje, es importante evaluar lo bien que el algoritmo aprendió de su experiencia. Para evaluar la *precisión del modelo*:

- Se utiliza un conjunto de datos de prueba.
- Se debe desarrollar medidas de rendimiento específicas para la aplicación prevista.

5. Mejorar el rendimiento del modelo: Si se necesita un mejor rendimiento, puede ser necesario cambiar a otro tipo de modelo. Puede que necesite complementar sus datos con otros datos adicionales.

Una vez completados estos pasos, si el modelo parece funcionar satisfactoriamente, se puede usarlo para proporcionar datos de puntuación para predicciones, proyecciones de datos financieros, y generar información útil para la investigación.

Los éxitos y fracasos del modelo realizazdo pueden informar sobre datos adicionales para realizar la siguiente generación de su modelo.

Elegir un algoritmo de aprendizaje automático

El proceso de selección de un algoritmo de aprendizaje automático implica hacer coincidir las características de los datos que se van a aprender con los sesgos de los métodos disponibles. Dado que la elección del algoritmo de aprendizaje automático depende en gran medida del tipo de datos que se analizarán y la tarea en cuestión, a menudo es útil pensar en este proceso a medida que recopila, explora y limpia los datos.

Pensando en los datos de entrada

Todos los algoritmos de aprendizaje automático requieren datos de entrenamiento de entrada, ya que, estos toman la forma de ejemplos y características.

Las características vienen en varias formas. Si una característica está medida en números se llama numérica. Si mide un atributo representado por un conjunto de categorías, se denomina categórica o nominal. Un caso especial de variables categóricas se llama ordinal, que designa una variable nominal con categorías que caen en una lista ordenada. Es importante considerar qué representan las características porque el tipo y la cantidad de características en su conjunto de datos ayudará a determinar un algoritmo de aprendizaje automático apropiado para su tarea.

Pensar en tipos de aprendizaje automático algoritmos

Los algoritmos de aprendizaje automático pueden dividirse en:

- ***Supervisados:*** Se utilizan para construir modelos predictivos.

  Un modelo predictivo se utiliza para tareas que implican la predicción de un valor utilizando otros valores del conjunto de datos.

  El algoritmo de aprendizaje intenta el algoritmo de aprendizaje de datos, intenta optimizar el modelo para que los valores de las características den como resultado, lo que se desea. El proceso de entrenamiento de un modelo predictivo se denomina aprendizaje supervisado.

  La característica objetivo a predecir es una característica categórica conocida como clase y se divide en categorías denominadas niveles.

  Una clase puede tener dos o más niveles, los cuales no tienen que ser ordinales. La clasificación se utiliza mucho, y es por eso que existen muchos tipos de algoritmos de clasificación.

  Para predecir estos valores numéricos la forma de predicción se basa en los modelos de regresión lineal a los datos de entrada.

  Los métodos de regresión se utilizan mucho para hacer previsiones, ya que cuantifican en términos exactos la asociación entre los insumos y el objetivo.

- ***No supervisados:*** Usados para construir modelos descriptivos.

  Un modelo descriptivo se utiliza para tareas que se beneficiarían de la información obtenida al resumir los datos de formas nuevas e interesantes, ninguna característica es más importante que otra.

La tarea de modelado descriptivo denominada *descubrimiento de patrones* se utiliza para identificar asociaciones frecuentes en los datos. Puede detectar patrones de comportamiento fraudulento, defectos genéticos o prevenir actividades delictivas, defectos genéticos o prevenir actividades delictivas.

Esta tarea consiste en dividir un conjunto de datos en grupos homogéneos. La máquina es capaz de identificar los grupos, es por eso que se requiere la intervención humana para interpretarlos.

Hacer coincidir sus datos con un algoritmo apropiado

La siguiente tabla enumera los tipos generales de algoritmos de aprendizaje automático. El aprendizaje de estos métodos proporcionará una base suficiente para dar sentido a otros métodos.

Model	Task
Supervised Learning Algorithms	
Nearest Neighbor	Classification
naive Bayes	Classification
Decision Trees	Classification
Classification Rule Learners	Classification
Linear Regression	Numeric prediction
Regression Trees	Numeric prediction
Model Trees	Numeric prediction
Neural Networks	Dual use
Support Vector Machines	Dual use
Unsupervised Learning Algorithms	
Association Rules	Pattern detection
k-means Clustering	Clustering

Para hacer coincidir una tarea de aprendizaje con un enfoque de aprendizaje automático, existen cuatro tipos de tareas: clasificación, predicción numérica, detección de patrones o agrupación. Ciertas tareas simplifican la elección del algoritmo.

Para la clasificación, es útil considerar las diversas distinciones entre los algoritmos. Los árboles de decisión dan como resultado modelos que se entienden fácilmente, mientras que los modelos de redes neuronales son notoriamente difíciles de interpretar.

Uso de R para el aprendizaje automático

Una gran comunidad de expertos que contribuyeron al software agregaron los algoritmos necesarios para el aprendizaje automático, denominados paquete. Existen paquetes gratuitos para cada uno de los algoritmos de aprendizaje automático.

Instalación y carga de paquetes R

A pesar de la gran cantidad de complementos de R disponibles, el formato del paquete hace que la instalación y el uso sean sencillos.

La forma más directa de instalar un paquete es a través de la función install.packages():

```
> install.package("package name")
```

R luego se conectará a CRAN y descargará el paquete en el formato correcto para su sistema operativo. Algunos paquetes, requieren paquetes adicionales (dependencias). De manera predeterminada, el instalador descargará e instalará automáticamente las dependencia.

La función de instalación también proporciona opciones adicionales para instalar desde un archivo local, instalar desde la fuente o usar versiones experimentales.

Instalación de un paquete mediante la interfaz point-and-click

Como alternativa al comando install.packages(), R proporciona una interfaz gráfica de usuario (GUI) para la instalación de paquetes.

En un sistema Microsoft Windows se puede acceder desde el comando install packages en el menú Paquetes. En Mac OS X, el comando se denomina Instalador de paquetes y se encuentra en el menú Paquetes y datos.

En Windows, aparecerá lista de paquetes, y solo se tiene que desplazar al paquete necesitado y hacer clic en el botón Aceptar para instalar el paquete y todas lasdependencias.

En Mac OS X, el menú de instalación del paquete ofrece opciones adicionales. Para cargar la lista de paquetes, se tiene que dar click en Obtener lista. Desplácese hasta el paquete y haga clic en Instalar, teniendo en cuenta que tienen que activar las casillas dependencias.

Cargando un paquete R

Para ahorrar memoria, R no carga todos los paquetes instalados de forma predeterminada, se debe usar la función library(). Para ser precisos, library se refiere a la ubicación donde está instalado el paquete, no al paquete en sí.

Para cargar el paquete que instalamos, debe escribir lo siguiente:

```
>library (package)
```

Capítulo 2

Gestión y comprensión de los datos

Uno de los primeros componentes clave de cualquier proyecto de aprendizaje automático es la gestión y comprensión de los datos recopilados. El algoritmo de aprendizaje es bueno si los datos de entrada son buenos y, en muchos casos, los datos de entrada son complejos, desordenados y poco fiables, es por eso que la mayor parte del esfuerzo invertido en proyectos de aprendizaje automático se dedica a la preparación y exploración de los datos.

Estructuras de datos en R

Existen numerosos tipos de estructuras de datos en los lenguajes de programación.

Dado que R es un lenguaje de programación es utilizado para el análisis estadístico de datos. Las estructuras de datos utilizadas con mayor frecuencia en el aprendizaje automático son vectores, factores, listas, matrices y marcos de datos, los cuales están especializado para una tarea.

Vectores

La estructura de datos fundamental de R es el vector, que almacena un conjunto de valores llamados elementos. Todos los elementos deben ser del mismo tipo.

Tipos de vectores:

- Entero (números sin decimales)
- Numérico (números con decimales)
- Carácter (datos de texto)
- Lógico (valores TRUE o FALSE).

- NULL, se utiliza para indicar la ausencia de cualquier valor
- NA, que indica un valor omitido.

Para crear vectores se usa la función de combinación c(), también se puede dar un nombre al vector utilizando el operador de flecha <-, que es el operador de asignación de R.

Por ejemplo, vamos a construir un vector. Se crea un un vector de caracteres llamado **nombre_sujeto,** un vector numérico llamado **temperatura** y un vector lógico **estado_gripe**; TRUE si tiene gripe, FALSE en caso contrario.

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

Se puede acceder a los registros contando el número del elemento en el conjunto, y rodeándolo con corchetes después del nombre del vector.

```
temperature[1]
```

```
[1] 98.1
```

Un rango de valores se puede obtener utilizando el operador dos puntos.

```
temperature[2:3]
```

```
[1]  98.6 101.4
```

Los elementos pueden excluirse especificando un número de elemento negativo.

```
temperature[-2]
```

```
[1]  98.1 101.4
```

Por último, es útil especificar un vector lógico que indique si cada valor debe incluirse.

```
temperature[c(TRUE, TRUE, FALSE)]
```

```
[1] 98.1 98.6
```

Factores

Son más eficientes que los vectores de caracteres porque las etiquetas de categoría se almacenan solo una vez (ahorra memoria). La codificación de variables categóricas como factores garantiza que el modelo tratará estos datos de forma adecuada.

Para crear un factor a partir de un vector de caracteres, simplemente aplique la función factor().

```
gender <- factor(c("MALE", "FEMALE", "MALE"))
gender
```

```
[1] MALE   FEMALE MALE
Levels: FEMALE MALE
```

Podemos agregar niveles adicionales que pueden no aparecer en los datos. Supongamos que agregamos otro factor para el tipo de sangre:

```
blood <- factor(c("O", "AB", "A"),
levels = c("A", "B", "AB", "O"))
blood
```

```
[1] O  AB A
Levels: A B AB O
```

Tenga en cuenta que cuando definimos el factor sanguíneo para los tres pacientes, especificamos un vector adicional de cuatro posibles tipos de sangre usando la instrucción level =. Como resultado, aunque nuestros datos incluyen solo los tipos O, AB y A, los cuatro tipos se almacenan con el factor sanguíneo como lo indican los niveles de salida: A B AB O. El almacenamiento del nivel adicional permite la posibilidad de agregar datos con el otro tipo de sangre en el futuro.

Listas

Se utiliza para almacenar un conjunto ordenado de valores, y permite diferentes tipos de valores.

```
subject_name[1]
```

```
[1] "John Doe"
```

```
temperature[1]
```

```
[1] 98.1
```

```
flu_status[1]
```

```
[1] FALSE
```

```
gender[1]
```

```
[1] MALE
Levels: FEMALE MALE
```

Una lista se crea utilizando la función list(), se tiene la opción de proporcionar nombres para cada valor en la secuencia de elementos. Los nombres permiten acceder posteriormente a los valores de la lista por su nombre, en lugar de por su posición numerada.

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

Se puede acceder a una lista utilizando los mismos métodos que a un vector, los nombres dan una claridad adicional para acceder a los valores.

```
subject1[2]
```

```
$temperature
[1] 98.1
```

A menudo es más fácil acceder directamente a la temperatura, añadiendo un $ y el nombre del valor

```
subject1$temperature
```

```
[1] 98.1
```

Es posible obtener varios elementos de una lista especificando un vector de nombres:

```
subject1[c("temperature", "flu_status")]
```

```
$temperature
[1] 98.1

$flu_status
[1] FALSE
```

Marco de Datos (Data Frames)

Puede entenderse como una lista de vectores o factores, cada uno de los cuales tiene exactamente el mismo número de valores.

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

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

Incluimos un parámetro adicional: stringsAsFactors = FALSE. Si no especificamos esta opción, R convertirá automáticamente cada vector de caracteres en un factor. Aquí, el campo subject_name no es un dato categórico; los nombres no son categorías de valores. Por lo tanto, establecer la opción stringsAsFactors en FALSE nos permite convertir a factores solo donde tiene sentido para el proyecto.

Cuando mostramos el marco 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
1     John Doe        98.1      FALSE   MALE     O
2     Jane Doe        98.6      FALSE FEMALE    AB
3 Steve Graves       101.4       TRUE   MALE     A
```

Un marco de datos tiene dos dimensiones y, por lo tanto, se muestra en formato de matriz. En términos de aprendizaje automático, las columnas son las características y las filas son los ejemplos.

Para extraer columnas enteras (vectores) de datos, la forma más directa de extraer un solo elemento, es referirse a él por su nombre.

```
pt_data$subject_name
```

```
[1] "John Doe"     "Jane Doe"     "Steve Graves"
```

También similar a las listas, se puede usar un vector de nombres para extraer varias columnas de un marco de datos:

```
pt_data[c("temperature", "flu_status")]
```

```
  temperature flu_status
1        98.1      FALSE
2        98.6      FALSE
3       101.4       TRUE
```

También puede ingresar pt_data[2:3] para extraer las columnas de temperatura y flu_status, pero enumerar las columnas por nombre da como resultado un código R claro y fácil de mantener.

Por ejemplo, para extraer el valor de la primera fila y la segunda columna (temperatura de John Doe), ingresaría:

```
pt_data[1, 2]
```

```
[1] 98.1
```

Si desea más de una fila o columna de datos, puede hacerlo especificando vectores. La siguiente declaración extraerá datos de las filas 1 y 3, y de las columnas 2 y 4:

```
pt_data[c(1, 3), c(2, 4)]
```

```
  temperature gender
1        98.1   MALE
3       101.4   MALE
```

Para extraer todas las filas o columnas, deje en blanco la parte de la fila o la 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
1     John Doe        98.1      FALSE   MALE     O
```

Para extraer todo:

```
pt_data[ , ]
```

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

Se puede acceder a las columnas por nombre en lugar de por posición, y se pueden usar signos negativos para excluir filas o columnas de datos. Por lo tanto, la declaración:

```
pt_data[c(1, 3), c("temperature", "gender")]
```

```
  temperature gender
1        98.1   MALE
3       101.4   MALE
```

Es equivalente a:

```
pt_data[-2, c(-1, -3, -5)]
```

```
  temperature gender
1        98.1   MALE
3       101.4   MALE
```

Matrices y Arreglos

Una matriz es una estructura de datos que representa una tabla bidimensional, con filas y columnas de datos. Pueden contener cualquier tipo de datos, aunque se usan con mayor frecuencia para operaciones matemáticas por lo que almacenan datos numéricos.

Para crear una matriz, proporcione 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 de 2x2 que almacene las primeras cuatro letras del alfabeto, podemos usar el parámetro nrow para solicitar que los datos se dividan en dos filas:

```
m <- matrix(c('a', 'b', 'c', 'd'), nrow = 2)
m
```

```
     [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
```

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

```
 m <- matrix(c('a', 'b', 'c', 'd'), ncol = 2)
m
```

```
     [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
```

Con seis valores, solicitar dos filas crea una matriz con tres columnas:

```
m <- matrix(c('a', 'b', 'c', 'd', 'e', 'f'), nrow = 2)
m
```

```
     [,1] [,2] [,3]
[1,] "a"  "c"  "e"
[2,] "b"  "d"  "f"
```

Solicitar dos columnas crea una matriz con tres filas:

```
m <- matrix(c('a', 'b', 'c', 'd', 'e', 'f'), ncol = 2)
m
```

```
     [,1] [,2]
[1,] "a"  "d"
[2,] "b"  "e"
[3,] "c"  "f"
```

Los valores de las matrices se pueden extraer utilizando la notación [fila, columna]. Se pueden solicitar filas o columnas enteras:

```
m[1, ]
```

```
[1] "a" "d"
```

```
m[, 1]
```

```
[1] "a" "b" "c"
```

El arreglo es una tabla de datos multidimensional. Tiene filas, columnas y cualquier cantidad de capas adicionales de valores

Gestión de datos con R

Uno de los retos a los que se enfrenta el trabajo con conjuntos de datos masivos es la recopilación de datos procedentes de diversas fuentes.

Guardar y cargar estructuras de datos de R

Para guardar una estructura de datos particular en un archivo que pueda ser recargado más tarde o transferir a otro sistema, puede utilizar la función save().

Si tuviéramos tres objetos llamados x, y, y z, podríamos guardarlos en un archivo mydata.RData.

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

El comando load() recreará cualquier estructura de datos ya guardada que estuviera en un fichero .RData. Para cargar el archivo mydata.RData que guardamos en el código anterior, escriba:

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

Si necesita terminar su sesión de R, el comando save.image() escribirá toda su sesión en un archivo de forma sencilla.

Importar y guardar datos de archivos CSV

Es muy común que los datos disponibles públicamente se almacenen en archivos de texto. Los cuales se pueden leer en cualquier computadora o sistema operativo. También se pueden exportar e importar desde/hacia programas como Microsoft Excel.

Un archivo de datos tabular ("tabla") está estructurado en forma de matriz. Los valores de características en cada línea están separados por un símbolo predefinido (delimitador). La primera línea de un archivo de datos tabulares enumera los nombres de las columnas de datos (línea de encabezado).

El formato de archivo de texto tabular más común es el archivo de valores separados por comas (CSV), que utiliza la coma como delimitador. Un ejemplo de archivo CSV es:

```
subject_name,temperature,flue_status,gender,blood_type
```

```
Steve Graves,101.4,TRUE,MALE,A

Jane Doe,98.6,FALSE,FEMALE,AB

John Doe,98.1,FALSE,MALE,O
```

Para cargar este archivo CSV en R, se usa read.csv() de la siguiente manera:

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

Necesitamos usar el parámetro stringsAsFactors = FALSE para evitar que R convierta todas las variables de texto en factores.

Si sus datos residen fuera del directorio de trabajo de R, puede especificar la ruta, por ejemplo, /ruta/a/misdatos.csv al llamar a la función read.csv().

R asume que el archivo CSV incluye una línea de encabezado. En caso de no tenerlo, especifique la opción header = FALSE y R asignará nombres de funciones predeterminados en la forma V1, V2, como se muestra:

```
mydata <- read.csv("usedcars.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, como el valor separado por tabuladores (TSV).

Para guardar un marco de datos en un archivo CSV, utilice la función write.csv(). Si su marco de datos se llama pt_data, simplemente ingrese:

```
write.csv(pt_data, file = "usedcars.csv")
```

Esto escribirá un archivo CSV con el nombre pt_data.csv en la carpeta de trabajo de R.

Importación de datos de bases de datos SQL

Si sus datos están almacenados en una base de datos ODBC (Open Database Connectivity) SQL (Structured Query Language) como Oracle, MySQL, PostgreSQL, Microsoft SQL o SQLite, el paquete RODBC se puede usar para importar estos datos directamente a una trama de datos R.

ODBC es un protocolo estándar para conectarse a bases de datos independientemente del sistema operativo o DBMS (Sistema de gestión de bases de datos).

Si aún no lo ha hecho, deberá instalar y cargar el paquete RODBC:

```
> install.packages ("RODBC")
```

```
> library (RODBC)
```

Abrir una conexión llamada mydb a la base de datos con el DSN my_dsn:

```
> mydb <- odbcConnect("my_dsn")
```

Si su conexión ODBC requiere un nombre de usuario y una contraseña, deben especificarse:

```
> mydb <- odbcConnect ("my_dsn", uid = "username", pwd = "pasword")
```

Podemos usar la función sqlQuery() para crear un marco de datos R a partir de las filas de la base de datos extraídas por consultas SQL. Esta función permite especificar stringsAsFactors = FALSE.

```
> patient_query <- "select * from patient_data where alive = 1"
```

```
> patient_data <- sqlQuery (channel = mydb, query = patient_query, stringsAsFactors
= FALSE)
```

La variable de datos_paciente será un marco de datos con todas las filas seleccionadas mediante la consulta SQL almacenada en consulta_paciente.

Cuando haya terminado de usar la base de datos, la conexión se puede cerrar:

```
> odbcClose(mydb)
```

Esto cerrará la conexión mydb. Aunque R cerrará automáticamente las conexiones ODBC al final de una sesión de R, es una mejor práctica hacerlo explícitamente.

Explorar y comprender los datos

Tras recopilar y cargar los datos, el siguiente paso en el proceso de aprendizaje automático consiste en examinar los datos en detalle.

Cuanto mejor comprenda sus datos mejor podrá adaptar un modelode aprendizaje automático a su problema de aprendizaje.

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

La exploración de datos es un proceso fluido, los pasos se pueden imaginar como una especie de investigación en la que se responde a preguntas sobre los datos, estos pasos básicos de esta investigación se deberian aplicar a cualquier conjunto de datos que desee.

Explorando la estructura de los datos

La función str() proporciona un método para mostrar la estructura de un marco de datos o cualquier estructura de datos R, incluidos vectores y listas.

```
str(usedcars)
```

```
'data.frame':   150 obs. of  19 variables:
 $ X.12        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.11        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.10        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.9         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.8         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.7         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.6         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.5         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.4         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.3         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.2         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X.1         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ 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. El número de observaciones a menudo se abrevia simplemente como n.

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. Mirando la línea de la característica llamada color, notamos algunos detalles adicionales:

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

Después del nombre de la variable, chr nos dice que la función es de tipo carácter. En este conjunto de datos, tres de las variables son de carácter, mientras que tres se indican como int. Aunque este conjunto de datos incluye solo variables de caracteres y enteros, también es probable que encuentre num cuando utilice datos que no sean enteros. Cualquier factor se enumeraría como Tipo de factor.

A veces, los conjuntos de datos tienen características con nombres y códigos sin sentido o simplemente un número como V1. Puede ser necesario realizar investigaciones adicionales para determinar qué representa realmente una característica.

Exploración de variables numéricas

Para investigar las variables numéricas de los datos, la función summary() muestra varios estadísticos de resumen comunes.

```
summary(usedcars$year)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2000    2008    2009    2009    2010    2012
```

También podemos utilizar 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.: 55124
 Max.   :21992   Max.   :151479
```

Los seis estadísticos de resumen que proporciona la función summary() son herramientas sencillas, pero potentes para investigar los datos. Los estadísticos de resumen pueden dividirse 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 usadas para identificar un valor que se encuentra en el medio de un conjunto de datos. Cuando algo se considera promedio, cae en algún lugar entre los extremos de la escala. En estadística, el promedio también se conoce como la media, una medida definida como la suma de todos los valores dividida por el número de valores.

```r
(36000 + 44000 + 56000) / 3
```

```
[1] 45333.33
```

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

```r
mean(c(36000, 44000, 56000))
```

```
[1] 45333.33
```

Aunque la media es la estadística más citada para medir el centro de un conjunto de datos, no siempre es la más adecuada. Otra medida de tendencia central de uso común es la mediana, que es el valor que se encuentra en la mitad de una lista ordenada de valores. R proporciona una función mediana():

```r
median(c(36000, 44000, 56000))
```

```
[1] 44000
```

Nota: Si un conjunto de datos tiene un número par de valores, no hay un valor medio. En este caso, la mediana suele calcularse como el promedio de los dos valores en el centro de la lista ordenada.

La media y la mediana se ven afectadas de manera diferente por los valores que caen en los extremos del rango. La media es muy sensible a los valores atípicos (altos o bajos), por lo que es más probable que se desplace hacia arriba o hacia abajo por un pequeño número de valores extremos.

Medición de la dispersión: los cuartiles y el resumen de cinco números

Conocer la dispersión, nos da una idea de los máximos y mínimos de los datos y de si la mayoría de los valores se parecen o no a la media y la mediana.

Conjunto de cinco estadísticos que representan aproximadamente la dispersión de un conjunto de datos:

1. Mínimo (Min.)
2. Primer cuartil, o Q1 (1er Qu.)

3. Mediana, o Q2 (Median)
4. Tercer cuartil, o Q3 (3rd Qu.)
5. Máximo (Max.)

El mínimo y el máximo son los valores más extremos del conjunto de datos, para encontrar estos valores se proporciona las funciones min() y max().

En R la función range() devuelve tanto el valor mínimo como el máximo. Combinando range() con la función de diferencia, diff() permite examinar el rango de datos con un solo comando:

```
range(usedcars$price)
```

```
[1]  3800 21992
```

```
diff(range(usedcars$price))
```

```
[1] 18192
```

El primer y tercer cuartil, Q1 y Q3, se refieren al valor por debajo o 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 el mismo número de valores.

El 50% medio de los datos entre Q1 y Q3 es de especial interés porque es una medida sencilla de la dispersión. La diferencia entre Q1 y Q3 se conoce como rango intercuartílico (IQR), y puede calcularse con la función IQR():

```
IQR(usedcars$price)
```

```
[1] 3909.5
```

La función quantile() es una herramienta para identificar cuantiles para un conjunto de valores, por defecto, devuelve el resumen de cinco números. Permite especificar entre nueve algoritmos diferentes especificando el parámetro type.

```
quantile(usedcars$price)
```

```
     0%     25%     50%     75%    100%
3800.0 10995.0 13591.5 14904.5 21992.0
```

Si especificamos un parámetro probs adicional con un vector que denota puntos de corte, podemos obtener cuantiles arbitrarios:

```
quantile(usedcars$price, probs = c(0.01, 0.99))
```

```
     1%      99%
5428.69 20505.00
```

La función de secuencia seq() se utiliza para generar vectores de valores espaciados uniformemente.

```
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
```

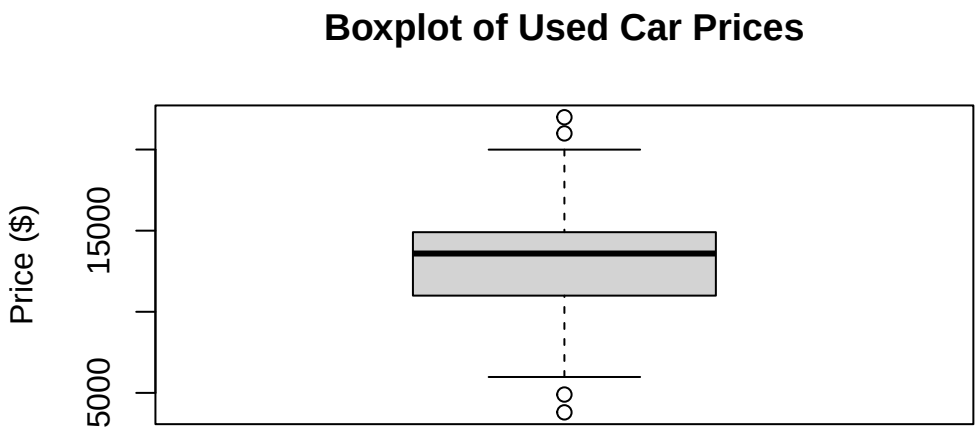
Una vez comprendido el resumen de cinco números, podemos volver a examinar.

Visualización de variables numéricas: diagramas de caja

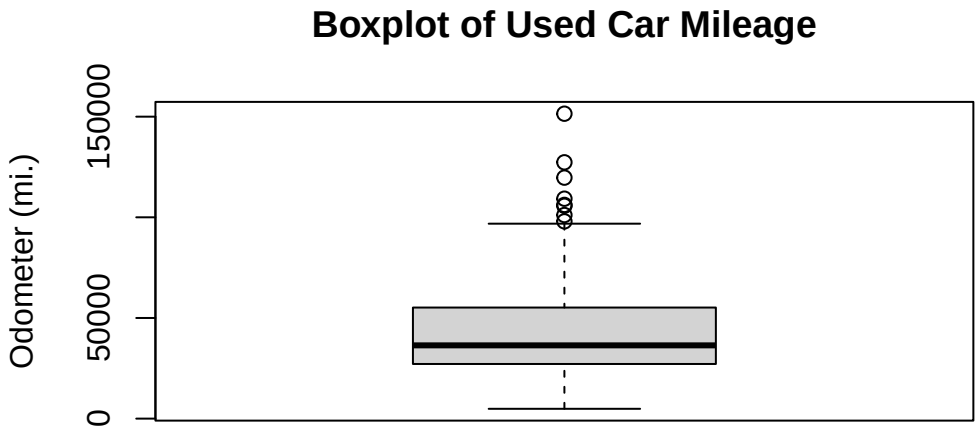
La visualización de variables numéricas puede ser útil para diagnosticar muchos problemas con los datos. El gráfico de caja muestra el centro y la dispersión de una variable numérica en un formato que le permite obtener rápidamente una idea del rango y el sesgo de una variable, o compararla con otras variables.

Para obtener un diagrama de caja para una variable, usaremos la función boxplot(). Se especifican un parámetros adicionales, main e ylab, para agregar el título a la figura y etiquetar el eje y:

```
boxplot(usedcars$price, main="Boxplot of Used Car Prices",
ylab="Price ($)")
```




```
boxplot(usedcars$mileage, main="Boxplot of Used Car Mileage",
ylab="Odometer (mi.)")
```



El diagrama de caja y patillas representa los valores de resumen de cinco números usando líneas horizontales. Las líneas horizontales que forman el cuadro en el medio de cada figura representan Q1, Q2 y Q3 cuando se lee el gráfico de abajo hacia arriba.

Nota: En diagramas de caja simples, el ancho de la caja y los bigotes es arbitrario y no ilustra ninguna característica de los datos. Es posible utilizar la forma y el tamaño de los recuadros para facilitar las comparaciones de los datos entre varios grupos.

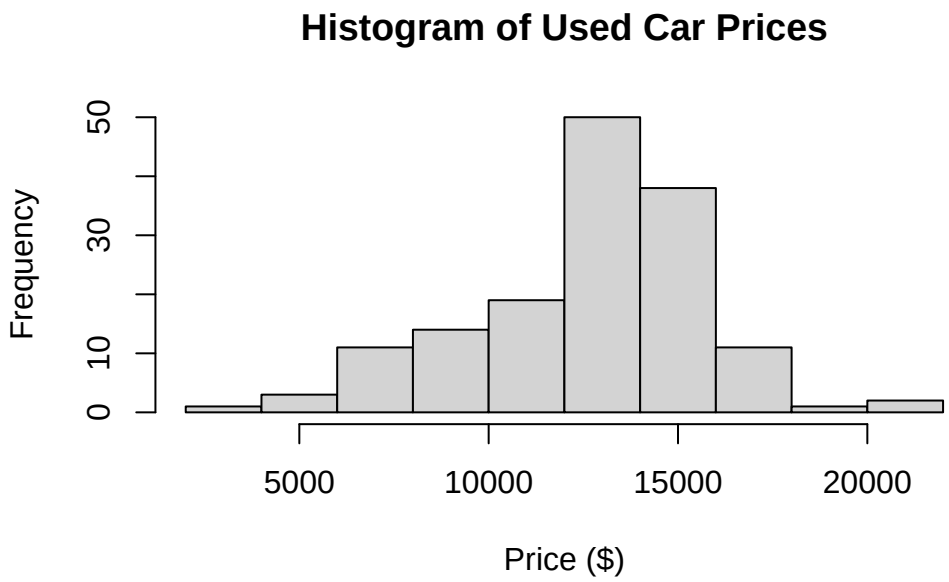
El mínimo y el máximo se ilustran con los bigotes que se extienden por debajo y por encima de la caja; sin embargo, es una convención permitir que los bigotes se extiendan hasta un mínimo o un máximo de 1,5 veces el IQR por debajo de Q1 o por encima de Q3. Los valores que superan este umbral se consideran valores atípicos y se indican como círculos o puntos.

Visualización de variables numéricas – Histogramas

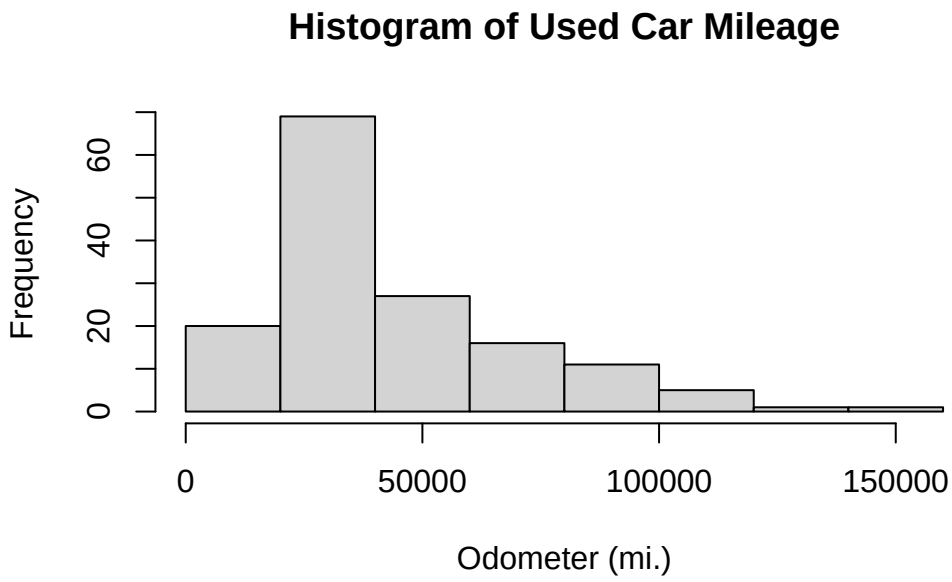
Un histograma representa gráficamente la dispersión de una variable numérica. Divide los valores de la variable en un número predefinido de porciones que actúan como contenedores de valores. Utiliza cualquier número de contenedores de idéntico ancho, pero permite estos contengan diferentes números de valores.

Podemos crear un histograma usando la función hist():

```
hist(usedcars$price, main = "Histogram of Used Car Prices",
xlab = "Price ($)")
```




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



El histograma se compone de una serie de barras con alturas que indican la frecuencia de los valores que caen dentro de cada uno de los contenedores. Las líneas verticales que separan las barras, (eje horizontal), indican los puntos inicial y final del rango de valores del contenedor.

Los histogramas de datos sesgados se ven estirados en uno de los lados:

La capacidad de diagnosticar rápidamente dichos patrones en nuestros datos es uno de los puntos fuertes del histograma como herramienta de exploración de datos.

Comprensión de datos numéricos - distribuciones uniforme y distribuciones normales

Los histogramas, los gráficos de caja y los estadísticos que describen el centro y la dispersión permiten examinar la distribución de los valores de una variable.

Si todos los valores tienen la misma probabilidad de ocurrir, 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.

Es importante tener en cuenta que no todos los sucesos aleatorios son uniformes.

Medición de la dispersión: varianza y desviación estándar

Las distribuciones nos permiten caracterizar una gran cantidad de valores utilizando una menor cantidad de parámetros. La distribución normal se puede definir con dos datos: centro y dispersión. El centro se define por su valor medio y la dispersión se mide mediante la desviación estándar.

Para calcular la desviación estándar, primero obtenemos la varianza, (promedio de las diferencias al cuadrado entre cada valor y el valor medio). En notación matemática, la varianza se define mediante la siguiente fórmula:

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

La desviación estándar:

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

Para obtener la varianza y la desviación estándar en R se pueden utilizar las funciones var() y sd()

```
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 se distribuyen más ampliamente alrededor de la media. La desviación estándar indica, en promedio, cuánto difiere cada valor de la media.

La desviación estándar se puede usar para estimar rápidamente qué tan extremo es un valor dado bajo el supuesto de que proviene de una distribución normal.

Exploración de variables categóricas

El conjunto de datos de coches usados tenía tres variables categóricas: modelo, color y transmisión. Como utilizamos el parámetro stringsAsFactors = FALSE al cargar los datos, R las ha dejado como variables de carácter (chr) en lugar de convertirlas automáticamente en factores. La función table() puede utilizarse para generar tablas unidireccionales

```
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 la tabla enumera las categorías de la variable nominal y un recuento del número de valores que entran en esa categoría.

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

```
       SE       SEL       SES
0.5200000 0.1533333 0.3266667
```

Supongamos que queremos mostrar los resultados en porcentajes con un solo decimal:

```
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
```

Medición de la tendencia central: la moda

En términos estadísticos, la moda de una característica es el valor que ocurre con más frecuencia (medida de tendencia central). Se usa para datos categóricos, ya que la media y la mediana no están definidas para variables nominales.

Una variable puede tener más de una moda; una variable con una moda es unimodal, con dos modas es bimodal. Los datos que tienen múltiples modos se denominan multimodales.

Los modos se utilizan en un sentido cualitativo para obtener una comprensión de los valores importantes en un conjunto de datos. Sin embargo, sería peligroso poner demasiado énfasis en la moda ya que el valor más común no es necesariamente una mayoría.

Pensar en las modas como valores comunes nos permite aplicar el concepto de moda estadística a los datos numéricos. Sería poco probable tener una moda para una variable continua, ya que es posible que no se repitan dos valores. Puede ser útil considerar la moda cuando se exploran datos numéricos, particularmente para examinar si los datos son multimodales o no.

Exploración de las relaciones entre variables

Se han examinado las variables de una en una, calculando sólo estadísticas univariantes. Durante nuestra investigación, nos planteamos preguntas que no pudimos responder en ese momento:

- ¿Implican los datos de precios que estamos examinando sólo coches de clase económica, o también hay coches de lujo con mucho kilometraje?
- ¿Permiten las relaciones entre el modelo y el color determinar el tipo de coche que estamos examinando?

Este tipo de preguntas pueden abordarse analizando la relación entre dos variables. Las relaciones de más de dos variables se denominan relaciones multivariantes.

Examen de las relaciones - bidireccionales tabulaciones cruzadas

Para examinar una relación entre dos variables nominales, se utiliza una tabulación cruzada bidireccional.

Una tabulación cruzada es similar a un gráfico de dispersión en el sentido de que permite examinar cómo varían los valores de una variable por los valores de otra.

Existen varias funciones para producir tablas bidireccionales en R, incluida table(), la función CrossTable() es quizás la más fácil de usar, ya que presenta los porcentajes de fila, columna y en una sola tabla.

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

Simplifiquemos nuestro proyecto reduciendo el número de niveles de la variable color. Esta variable tiene nueve niveles, pero realmente no necesitamos tanto detalle.

Crearemos una variable indicadora binaria, la cual indica si el color del coche es conservador o no, su valor será 1 si es verdadero, 0 en caso contrario.

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

Puede que haya notado un nuevo comando aquí: el operador %in% devuelve TRUE o FALSE para cada valor en el vector a la izquierda del operador. En términos sencillos, puede traducir esta línea como "¿está el color del coche usado en el conjunto de negro, gris, plata y blanco?".

```
table(usedcars$conservative)
```

```
FALSE  TRUE
   51    99
```

Veamos ahora una tabulación cruzada para ver cómo varía la proporción de coches de color varía según el modelo, trataremos conservador como la variable dependiente (y).

El comando CrossTable() es por lo tanto:

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

Hay una gran cantidad de datos en la salida de CrossTable(). La leyenda en la parte superior indica cómo interpretar cada valor. Las proporciones indican la proporción de esa celda en relación con la estadística Chi-cuadrado, el total de la fila, el total de las columnas y el total de la tabla.

Los valores de Chi-cuadrado se refieren a la contribución de la celda en la prueba de Chi-cuadrado 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 fuerte evidencia de que las dos variables están asociadas.

Puede obtener los resultados de la prueba de chi-cuadrado agregando un parámetro adicional que especifique chisq = TRUE al llamar a la función CrossTable()