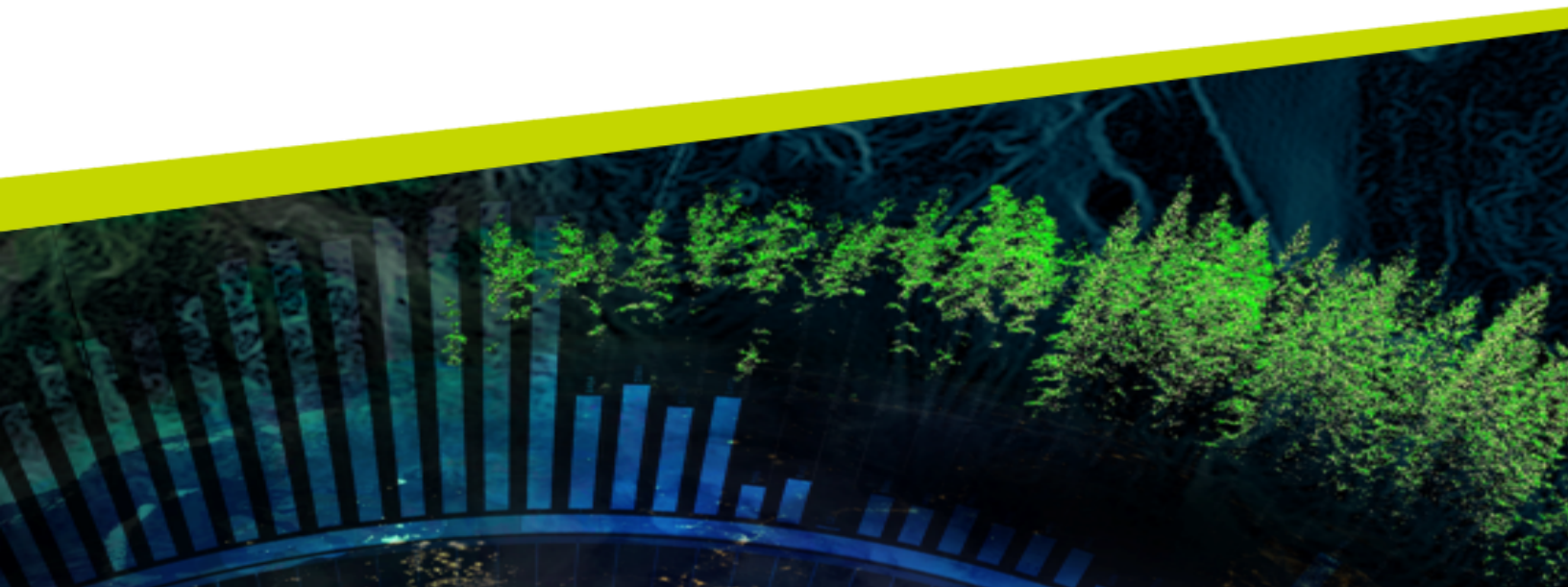
INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.



CAPÍTULO 1. INTRODUCCIÓN

Este libro tiene como propósito acompañarte en el aprendizaje de las primeras nociones de inferencia estadística y de creación de modelos estadísticos. En este primer capítulo comenzaremos por buscar definiciones iniciales para los conceptos de inferencia y modelo, para luego abordar algunas nociones iniciales acerca de los datos empleados en estadística y algunas herramientas para que puedas empezar a usar el entorno de programación R, con el cual trabajaremos a lo largo de todo el texto.

1.1 INFERENCIA Y MODELOS ESTADÍSTICOS

La Real Academia Española (2014) define **inferencia** como "acción y efecto de inferir". Esto por sí solo no nos dice mucho, pero si buscamos también la definición de **inferir**, encontraremos que significa "deducir algo o sacarlo como conclusión de otra cosa". A partir de estas definiciones, y de acuerdo con Devore (2008, p. 5), podemos decir que la **estadística inferencial** es una rama de la estadística que busca obtener una conclusión para un conjunto de individuos o elementos (denominado **población**) a partir de información recolectada de un subconjunto de éste (llamado **muestra**).

Llegar a una definición de **modelo estadístico** puede ser bastante más complejo. Como nos muestra la figura 1.1, ¡un modelo puede ser muchas cosas diferentes! Veamos qué nos dice la Real Academia Española (2014):

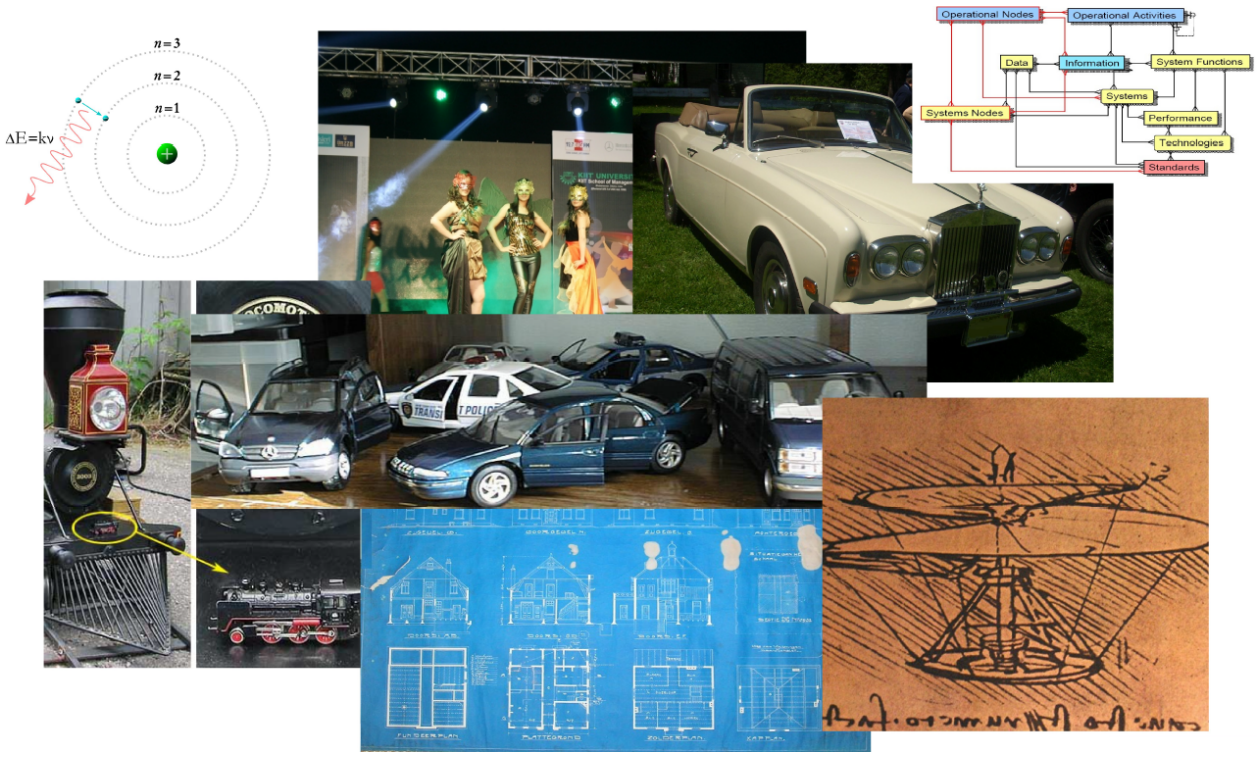


Figura 1.1: ejemplos de modelos.

1. Arquetipo o punto de referencia para imitarlo o reproducirlo.
2. En las obras de ingenio y en las acciones morales, ejemplar que por su perfección se debe seguir e imitar.
3. Representación en pequeño de alguna cosa.

4. Esquema teórico, generalmente en forma matemática, de un sistema o de una realidad compleja, como la evolución económica de un país, que se elabora para facilitar su comprensión y el estudio de su comportamiento.
5. Objeto, aparato, construcción, etc., o conjunto de ellos realizados con arreglo a un mismo diseño. *Auto modelo 1976. Lavadora último modelo.*
6. Vestido con características únicas, creado por determinado modista, y, en general, cualquier prenda de vestir que esté de moda.
7. En empresas, u. en aposición para indicar que lo designado por el nombre anterior ha sido creado como ejemplar o se considera que puede serlo. *Empresa modelo. Granjas modelo.*
8. Esc. Figura de barro, yeso o cera, que se ha de reproducir en madera, mármol o metal.
9. *Cuba* impreso (||hoja con espacios en blanco).
10. Persona que se ocupa de exhibir diseños de moda.
11. Persona u objeto que copia el artista.

Puede ser de ayuda tener en cuenta algunas definiciones e ideas que nos ofrece la literatura. Kaplan (2009), por ejemplo, señala que un modelo es una representación con un propósito particular. Pero otros contribuyen a enriquecer esta definición:

- Representación simplificada de la realidad en la que aparecen algunas de sus propiedades (Joly, 1988).
- Permiten estudiar de forma simple y comprensible una porción de la realidad (Ríos, 1995).
- Dejan cosas fuera y pueden llevar a conclusiones equivocadas (Kaplan, 2009).
- Resumen de manera conveniente, a juicio de los sus creadores, los aspectos más relevantes del fenómeno estudiado y sus relaciones (Mendez Ramírez, 1998).
- Están mediados por el diseño, es decir la forma de seleccionar y observar (o manipular) la realidad modelada (Mendez Ramírez, 2012).
- Son pequeños, económicos, seguros y fáciles de transportar, copiar y modificar (Kaplan, 2009).

Si a esta concepción del significado de la palabra modelo le agregamos nuevos conceptos fundamentales que nos ofrecen otros autores, podemos acercarnos un poco más a la idea de **modelo estadístico**:

- Modelo matemático de la regularidad estadística de los posibles resultados de la evolución de un fenómeno aleatorio (Mendez Ramírez, 1998).
- Distribución de probabilidad construida para poder hacer inferencias o tomar decisiones desde datos (Freedman, 2009).
- Descripción simple de un proceso probabilístico que puede haber dado origen a un conjunto de datos observados (McCullagh, 2002).
- Modelo estocástico que contiene parámetros desconocidos que deben ser estimados en base a suposiciones acerca del modelo y los datos (SAS Institute Inc., 2008).

Pero, ¿para qué sirven los modelos estadísticos? Diversos autores nos muestran que tales modelos son muy útiles en diversos contextos:

- Para describir (Kaplan, 2009) o resumir datos (Freedman, 2009).
- Para clasificar (Kaplan, 2009) o predecir (Freedman, 2009; Kaplan, 2009).
- Para anticipar el resultado de intervenciones (solo cuando el modelo es de tipo causal) (Freedman, 2009; Kaplan, 2009).

Ahora que hemos definido nuestros conceptos iniciales, veamos algunas definiciones y herramientas que nos permitan comenzar a explorarlos.

1.2 VARIABLES, PARÁMETROS Y ESTADÍSTICOS

Comencemos a partir de un ejemplo para ilustrar algunas ideas iniciales acerca de los datos. La tabla 1.1 muestra las primeras filas de la matriz de datos `ffaa`[1], que almacena información acerca de miembros activos de las Fuerzas Armadas de Chile. Cada **columna** de la matriz representa una **variable** o característica, mientras que cada **fila** corresponde a una **unidad de observación** o instancia. Así, cada fila de la tabla 1.1 almacena datos de una misma persona. El uso de **matrices de datos** para almacenar los datos es muy conveniente, pues nos ayuda a acceder a los datos y modificarlos más fácilmente. Por ejemplo, para agregar una nueva observación, basta con añadir una nueva fila a la matriz. Si queremos eliminar una característica, simplemente borramos la columna correspondiente. Para consultar una característica en particular de una observación, solo necesitamos conocer la fila y la columna correspondientes.

id	género	estatura	escalafón	servicio	antigüedad	rama
1	M	1,77	S	89,91	15	E
2	M	1,97	O	65,14	30	M
3	F	1,65	O	97,03	12	A
4	M	1,82	S	76,29	9	A
5	F	1,73	S	69,46	7	M
6	M	1,78	S	97,67	21	E
7	M	1,87	O	72,09	27	M
8	F	1,91	S	94,53	11	A
⋮	⋮	⋮	⋮	⋮	⋮	⋮
6051	M	1,72	S	86,48	17	E

Tabla 1.1: algunas filas de la matriz de datos `ffaa`.

Antes de comenzar a trabajar con los datos, tenemos que estar seguros de comprender cada uno de sus aspectos. Para ello, las siguientes preguntas pueden ser un excelente punto de partida:

- ¿A qué corresponde cada característica?
- ¿Cuáles son sus unidades de medición?
- ¿Qué valores puede tomar?

La tabla 1.2 muestra la descripción de las características presentes en la matriz de datos de la tabla 1.1. En ella se explica el significado de cada columna de la matriz junto al rango, o enumeración, de valores posibles. Notemos que esto último se omite para algunas variables, ya que el rango de valores posibles queda definido por su descripción. También que es importante señalar la unidad de medida cuando pueda haber ambigüedad, como se hace con la variable `estatura`.

Variable	Descripción
id	Identificador de la observación.
género	Género de la persona (M: masculino, F: femenino).
estatura	Estatura (m).
escalafón	Escalafón al que pertenece (O: oficial, S: suboficial).
servicio	Evaluación de servicio (entre 0 y 100).
antigüedad	Años que lleva en servicio activo.
rama	Rama de las FF.AA. a la que pertenece (E: Ejército, M: Marina, A: Fuerza Aérea).

Tabla 1.2: descripción de las variables para el conjunto de datos `ffaa`.

Si estudiamos la tabla 1.2 con detalle, podemos notar diferencias interesantes entre las variables, más allá de su descripción. Por ejemplo, no todas ellas pueden tomar los mismos valores. Con esto aparece la noción de **tipos de variables**, los cuales podemos jerarquizar:

[1] Los datos aquí presentados son ficticios y han sido creados únicamente con fines pedagógicos.

- **Numéricas**: pueden tomar muchos valores numéricos, y son sensibles a operaciones aritméticas. Pueden separarse en:
  - **Continuas**: pueden tomar cualquier valor (en un intervalo) del conjunto de los reales. Por ejemplo, las variables `estatura` y `servicio` descritas en la tabla 1.2.
  - **Discretas**: no es posible que tomen cualquier valor (en un intervalo). Por ejemplo, podrían tomar únicamente valores enteros no negativos, como la variable `antigüedad` de la matriz de datos `ffaa`.
- **Categóricas**: solo pueden tomar un valor de entre un conjunto acotado. Cada posible valor se denomina **nivel**. Entre las variables categóricas es posible distinguir variables:
  - **Nominales**: no existe un orden natural entre los niveles. Ejemplos de variables nominales son `género` y `rama` en la matriz de datos `ffaa`.
  - **Ordinales**: existe un orden natural entre los niveles. Por ejemplo, la idea de jerarquía es evidente al distinguir entre oficiales y suboficiales en la variable `escalafón` del ejemplo.

Tener diferentes tipos de variables significa que debemos medirlas con distintas clases de escalas, las cuales se distinguen por sus propiedades y los tipos de operaciones que permiten:

- **Escala nominal**: sirve solo para separar un conjunto de elementos en subclases excluyentes entre sí. Los valores no son más que nombres o estados, por lo que no podemos hacer operaciones aritméticas ni podemos establecer relaciones de orden.
- **Escala ordinal o de rangos**: esta escala, al igual que la nominal, permite separar un conjunto de elementos en subclases excluyentes entre sí. Una vez más, los valores son solo nombres o estados, por lo que tampoco podemos hacer operaciones aritméticas. Pero en este caso sí podemos establecer una relación de orden, que es más clara cuando la variable tiene a lo menos tres niveles. A modo de ejemplo, si queremos una variable para medir el nivel de estudios de las personas en un grupo demográfico, podríamos considerar una escala ordinal con los niveles "ninguna", "básica", "media", "superior" y "postgrado". Aquí podemos apreciar claramente que los niveles están ordenados de manera creciente considerando los años de estudios formales.
- **Escala de intervalos iguales**: sirve para datos continuos o discretos con una gran cantidad de niveles. Además de la noción de orden de la escala ordinal, se cumple que la distancia entre dos valores cualesquiera de la escala es conocida y constante, por lo que podemos emplear operaciones aritméticas. Aunque el punto cero y la unidad de medida son arbitrarios, la razón entre dos intervalos es independiente de ambos elementos. Tomemos, por ejemplo, la escala Celsius de temperatura, donde el valor cero está dado por el punto de congelación del agua y el valor 100 por el punto de su ebullición. Los valores intermedios de esta escala se calculan en base a estos puntos. Debemos notar que, a pesar de estos parámetros arbitrarios, el cambio en la cantidad de calor es el mismo si aumentamos la temperatura de 10 a 15 grados Celsius, que si la aumentamos de 25 a 30. Si miramos ahora la escala Fahrenheit de temperatura, los puntos fijos son diferentes a los empleados por la escala Celsius, por lo que el cero no significa lo mismo. Nuevamente un aumento de la temperatura de 10 a 15 grados Fahrenheit es el mismo aumento de calor medido por un incremento de 25 a 30 grados Fahrenheit. En ambas escalas, un intervalo del mismo tamaño, de 5 grados en el ejemplo, representa la misma diferencia de calor, independiente de los valores en los extremos del intervalo.
- **Escala de razón**: cumple con todos los atributos de la escala de intervalos, pero además tiene su origen en un cero verdadero. Ejemplos de tales escalas son las que permiten medir la masa o la distancia. En el ejemplo con las escalas de intervalos iguales vimos que un intervalo de 5 grados representa el mismo cambio de calor en cada escala de temperatura, sin embargo, un aumento de 5 grados Celsius no es equivalente a un aumento de 5 grados Fahrenheit (aunque existe una transformación lineal que nos permite transformar una medida en una escala a su equivalente en otra escala). En contraste, en una escala de razón, la diferencia entre dos puntos es independiente de la unidad de medida. Por ejemplo, si comparamos la masa de Júpiter con la masa de la Tierra, la razón de 317,8165 veces es constante e independientemente de si empleamos toneladas, kilogramos, libras u onzas.

La estadística usa los datos para responder diversas preguntas, muchas de las cuales se orientan a encontrar relaciones entre variables. Así, dos variables pueden ser:

1. **Independientes**: no existe asociación o relación entre las variables.
2. **Dependientes**: existe una asociación o relación entre las variables. Puede existir:
   - **Asociación positiva**: si una variable crece, la otra también lo hace.
   - **Asociación negativa**: si una variable crece, la otra decrece.

   Notemos eso sí, que esto no significa necesariamente que una variable es responsable del cambio de la variable relacionada.

En el contexto de la estadística, decimos que un **parámetro** es cualquier número que describa una población en forma resumida, como por ejemplo la media poblacional. A su vez, un **estadístico** es "cualquier cantidad cuyo valor puede ser calculado a partir de datos muestrales" (Devore, 2008, p. 204), como por ejemplo la media, la mediana o la desviación estándar de un conjunto de datos observados. Si bien a primera vista ambos conceptos parecen similares, en realidad existe una diferencia importante entre ellos: el parámetro describe una población, mientras que el estadístico, al ser calculado a partir de una muestra, no es más que una **estimación puntual** del parámetro.

Si necesitas más ejemplos o quieres complementar lo aprendido, puedes consultar los textos de referencia para esta sección. Diez, Barr y Çetinkaya-Rundel (2017, pp. 9-19) describe los principales conceptos relativos a datos, tipos de variables y relaciones entre variables. En Dagnino (2014) puedes aprender más sobre escalas de medición.

1.3 CONOCIENDO R

R es un ambiente de software gratuito para estadística computacional y elaboración de gráficos. En esta sección conoceremos algunas herramientas que nos ayudarán a lo largo de este libro. Desde luego, estas breves páginas no pretenden ser un tutorial completo del lenguaje, sino más bien un punto de partida para que podamos aplicar los contenidos que aquí se abordan. Como ya señalamos, sugerimos el uso del entorno integrado de desarrollo RStudio, cuya documentación e instrucciones de instalación podemos consultar en Posit Software PBC (2024). En The R Foundation (s.f.) y Carchedi, De Mesmaeker y Vannoorenberghe (s.f.) podemos encontrar documentación acerca del lenguaje R y sus paquetes.

1.3.1 Importación de datos

Una de las primeras cosas que necesitamos conocer es cómo importar o cargar una matriz de datos (denominada *data frame* en R) desde un archivo de texto plano (.txt) o de valores separados por coma (.csv). Para lograrlo con éxito, debemos tener en cuenta algunas orientaciones para preparar los datos adecuadamente:

- La primera fila se usa para los nombres de las columnas (variables).
- La primera columna contiene los nombres de las observaciones, que deben ser únicos.
- Los nombres de las columnas deben respetar las convenciones de R:
  - No está permitido el uso de espacios ni símbolos especiales (?, $, *, +, #, (,), -, /, }, {, |, >, <, etc.). Solo se admite el uso de puntos (.) y guiones bajos (_).
  - Los nombres de variables no pueden comenzar con un dígito.
  - Los nombres de las columnas deben ser únicos.
- R es sensible a las mayúsculas.
- No puede haber filas en blanco.
- No debe tener comentarios.
- Los valores faltantes deben ser denotados mediante `NA`.
- Para columnas con fechas, se usa el formato mm/dd/aaaa.

- El archivo debe tener tener uno de los siguientes formatos, ejemplificados en la figura 1.2:
  - Extensión .txt con tabulaciones como delimitador y punto decimal para valores flotantes.
  - Extensión .csv en formato inglés, con comas (`,`) como delimitador y punto decimal (`.`) para valores flotantes.
  - Extensión .csv en formato español, con punto y comas (`;`) como delimitador y coma decimal (`,`) para valores flotantes.

id	género	estatura	escalafón	servicio	antigüedad	rama
1	M	1.77	S	89.91	15	E
2	M	1.97	O	65.14	30	C
3	F	1.65	O	97.03	12	A
4	M	1.82	S	76.29	9	A
5	F	1.73	S	69.46	7	M
6	M	1.78	S	97.67	21	E
7	M	1.87	O	72.09	27	C
8	F	1.91	S	94.53	11	A

(a) Texto plano delimitado por tabulaciones.

```
id,género,estatura,escalafón,servicio,antigüedad,rama
1,M,1.77,S,89.91,15,E
2,M,1.97,O,65.14,30,C
3,F,1.65,O,97.03,12,A
4,M,1.82,S,76.29,9,A
5,F,1.73,S,69.46,7,M
6,M,1.78,S,97.67,21,E
7,M,1.87,O,72.09,27,C
8,F,1.91,S,94.53,11,A
```

(b) Valores separados por comas (inglés).

```
id;género;estatura;escalafón;servicio;antigüedad;rama
1;M;1,77;S;89,91;15;E
2;M;1,97;O;65,14;30;C
3;F;1,65;O;97,03;12;A
4;M;1,82;S;76,29;9;A
5;F;1,73;S;69,46;7;M
6;M;1,78;S;97,67;21;E
7;M;1,87;O;72,09;27;C
8;F;1,91;S;94,53;11;A
```

(c) Valores separados por punto y comas (español).

Figura 1.2: formatos de archivo para importar datos en R.

El script 1.1 muestra las diferentes funciones para importar datos en R, donde las líneas que comienzan por `#` corresponden a comentarios. La línea 2 carga el conjunto de datos `mtcars`, disponible en R, mientras que las líneas 5, 9 y 16 importan datos desde archivos. Tanto `read.delim()` como `read.csv()` y `read.csv2()` se usan de la misma forma, pudiendo recibir como argumento una llamada al selector de archivos (`file.choose()`), como en la línea 5, o la ruta completa para el archivo, como en la línea 9. En el caso de la línea 16, basta con proporcionar el nombre de archivo pues la función `setwd()` (línea 12) permite establecer el directorio de trabajo de R para la sesión. Las funciones `head()` y `tail()` (líneas 20 y 24) proporcionan una buena manera de inspeccionar los datos cargados, pues muestran por consola las primeras y últimas filas de la matriz de datos, respectivamente.

Script 1.1: sentencias para importar un conjunto de datos.

```
1  # Cargar un conjunto de datos disponible en R.
2  datos1 <- mtcars
3
4  # Importar desde un archivo de texto plano delimitado por tabuladores.
5  datos2 <- read.delim(file.choose())
6
7  # Importar desde un archivo de valores separados por coma
8  # en formato inglés (figura 1.2 b).
9  datos3 <- read.csv("C:\\Inferencia\\ejemplo1-csv-eng.csv")
10
11 # Configurar carpeta de trabajo
12 setwd("C:\\Inferencia")
13
14 # Importar desde un archivo de valores separados por coma
15 # en formato español (figura 1.2 c).
16 datos4 <- read.csv2("ejemplo1-csv-esp.csv")
17
18 # Mostrar las primeras 6 filas del conjunto de datos
19 # almacenado en datos1.
```

```r
head(datos1)

# Mostrar las últimas 6 filas del conjunto de datos
# almacenado en datos1.
tail(datos1)
```

1.3.2 Importación de paquetes

Si bien el entorno R básico incluye muchísimas funcionalidades, existe una enorme variedad de paquetes o colecciones que incorporan otras nuevas o mejoran las ya existentes.

Antes de usar un paquete por primera vez tenemos que instalarlo. Para ello, podemos usar la sentencia que se muestra en la línea 2 del script 1.2. Debemos tener en cuenta que la función `install.packages()` requiere que el nombre del paquete se escriba entre comillas.

Para poder usar un paquete, existen las sentencias `library()` (línea 5 del script 1.2) y `require()` (línea 8), que reciben como argumento el nombre del paquete (sin comillas). Si bien ambas sentencias pueden usarse indistintamente, se diferencian en que `library()` termina la ejecución con un mensaje de error si el paquete no está instalado, mientras que `require()` solo emite una advertencia.

Una forma elegante de evitar errores es verificar si un paquete se encuentra instalado antes de usarlo, para lo que podemos usar una combinación de las sentencias anteriores, como muestran las líneas 11 a 14 del script 1.2. Cabe destacar que la opción `dependencies = TRUE` en la línea 12 asegura que se instalen además aquellos paquetes que son requeridos por el que se desea instalar. Fijémonos que el lenguaje de programación R usa **argumentos con nombre**.

Script 1.2: instalar y cargar paquetes de R.

```r
# Instalar un paquete.
install.packages("ggpubr")

# Primera forma de importar un paquete.
library(ggpubr)

# Segunda forma de importar un paquete.
require(ggplot2)

# Importar un paquete, instalándolo de ser necesario.
if(!require(dplyr)){
  install.packages("dplyr", dependencies=TRUE)
  require(dplyr)
}
```

1.3.3 Construcción de una matriz de datos

Consideremos la idea de construir una matriz de datos que contenga el nombre, la fecha de nacimiento y las calificaciones de los estudiantes en las tres evaluaciones de una asignatura. El script 1.3 crea esta matriz de datos en R con tres observaciones. En las líneas 2 a 4 crea un vector de strings con los nombres de los estudiantes y lo almacena en la variable `nombre`. De manera similar, en la línea 8 crea un vector de fechas. Debemos notar que para ello construye un vector de tres strings con las fechas en formato `aaaa-mm-dd`, el cual es entregado como argumento a la función `as.Date()` para que sean convertidos al formato de fecha. Las líneas 12 a 14 crean tres vectores de valores flotantes para las calificaciones obtenidas por los estudiantes. Hasta este punto, solo se tienen muchas variables con vectores de largo 3, los cuales deben ser combinados para formar una matriz de datos donde cada vector sea una columna. La función `data.frame()`, en las líneas 18 a 22, realiza esta tarea. Dicha función recibe como argumentos tantos vectores como variables tenga el conjunto de datos, y toma los nombres de las variables que los contienen como nombres de las columnas. Cabe

destacar que, en la línea 23, `data.frame()` recibe un argumento adicional, el booleano `stringsAsFactors`, con valor falso. Esto se debe a que, si no se entrega este parámetro, R asume que su valor por defecto es verdadero, por lo que interpreta el vector de strings como una variable categórica y asigna un valor numérico a cada nivel.

La última línea del script 1.3 permite guardar la matriz de datos en un archivo de valores separados por comas (formato español). La función `write.csv2()` recibe como argumentos el nombre de la variable que contiene la matriz de datos y una cadena de caracteres con el nombre del archivo. El argumento `row.names = FALSE` indica que no deseamos guardar los nombres de las filas. Si queremos guardar nuestra matriz de datos en un archivo separado por comas en formato inglés, podemos hacerlo mediante la función `write.csv()`, que funciona del mismo modo que `write.csv2()`.

Script 1.3: construir una matriz de datos.

```
# Crear un vector de strings y guardarlo en la variable nombre.
nombre <- c("Alan Brito Delgado",
            "Zacarías Labarca del Río",
            "Elsa Payo Maduro")

# Crear un vector de fechas y guardarlo en la variable
# fecha_nacimiento.
fecha_nacimiento <- as.Date(c("2008-1-25", "2006-10-4", "2008-3-27"))

# Crear tres vectores de reales entre 1.0 y 7.0 y guardarlos
# en prueba_i, respectivamente.
prueba_1 <- c(5.5, 3.4, 4.5)
prueba_2 <- c(3.2, 4.7, 4.1)
prueba_3 <- c(4.8, 4.3, 5.1)

# Construir un data frame a partir de los vectores anteriores y
# guardarlo en la variable dataframe.
dataframe <- data.frame(nombre,
                        fecha_nacimiento,
                        prueba_1,
                        prueba_2,
                        prueba_3,
                        stringsAsFactors = FALSE)

# Guardar un dataframe en un archivo csv (formato español).
write.csv2(dataframe, "Ejemplo.csv", row.names = FALSE)
```

1.3.4 Modificación de una matriz de datos

Muchas veces tendremos la necesidad de modificar la matriz de datos. Algunas tareas, como agregar o quitar una columna o un observación pueden hacerse de manera bastante sencilla, como ilustra el script 1.4.

Script 1.4: modificaciones sencillas de una matriz de datos.

```
# Leer un dataframe desde archivo csv.
datos <- read.csv2("Ejemplo.csv", stringsAsFactors = FALSE)

# Eliminar del data frame la columna fecha_nacimiento.
datos$fecha_nacimiento <- NULL

# Agregar al data frame la columna edad.
datos$edad <- c(23, 25, 23)

# Crear una nueva observación.
nueva <- data.frame(nombre="Elba Calao del Río",
```

```
                    prueba_1 = 6.4,
                    prueba_2 = 2.3,
                    prueba_3 = 4.6,
                    edad = 24)

# Agregar la nueva observación al data frame.
datos <- rbind(datos, nueva)

# Eliminar las primeras 3 observaciones del data frame.
datos <- datos[-c(1:3),]

# Guardar el dataframe en un archivo csv .
write.csv2(datos, "Ejemplo_mod.csv", row.names = FALSE)
```

Sin embargo, también podemos vernos en la necesidad de realizar transformaciones más complejas. El paquete `dplyr` ofrece un conjunto de funciones que simplifica esta tarea:

- `filter()`: selecciona instancias (filas) de acuerdo al valor contenido en una o más variables.
- `arrange()`: modifica el orden de las filas.
- `select()`: permite seleccionar variables (columnas) por sus nombres, a la vez que las reordena.
- `mutate()`: permite cambiar los valores de una variable o agregar nuevas variables que se obtienen como funciones de otras ya existentes.

Para mostrar el uso de estas funciones (script 1.5) usaremos el conjunto de datos `iris`, disponible en R. Este contiene 150 observaciones pertenecientes a tres especies de una flor llamada iris: setosa, versicolor y virginica, para las cuales se registran el largo y ancho de sus sépalos y de sus pétalos (en centímetros). Puedes consultar otras funciones y ejemplos más detallados en Müller (2021) y Wickham y Grolemund (2017, cap. 5).

Script 1.5: modificación de una matriz de datos con el paquete `dplyr`.

```
library(dplyr)

# Cargar dataframe iris incluido en R.
datos <- iris

# Seleccionar observaciones correspondientes a la especie versicolor.
versicolor <- datos |> filter(Species == "versicolor")

# Seleccionar observaciones de la especie versicolor cuyos sépalos tengan una
# longitud igual o superior a 6 cm.
largas <- datos |> filter(Species == "versicolor" & Sepal.Length >= 6)

# Seleccionar la especie y variables relativas a los pétalos.
petalos <- datos |> select(Species, starts_with("Petal"))

# Seleccionar variables de ancho y la especie.
anchos <- datos |> select(ends_with("Width"), Species)

# Agregar al conjunto de datos de los pétalos una nueva variable con la razón
# entre el largo y el ancho de éstos.
petalos <- petalos |> mutate(Species, Petal.Width,
                             Petal.Ratio = Petal.Length / Petal.Width)

# Ordenar el conjunto de datos de pétalos en forma descendente según la razón
# de los pétalos.
petalos <- petalos |> arrange(desc(Petal.Ratio))

# Ordenar el conjunto de datos de pétalos en forma ascendente según el largo de
```

```
# los pétalos.
petalos <- petalos |> arrange(Petal.Length)
```

En el script 1.5 aparece frecuentemente el operador `|>`, llamado *pipe*, cuya función es entregar un valor o el resultado de una expresión a la siguiente llamada a una función. En términos sencillos, la expresión `x |> f` es equivalente a `f(x)`, y su utilidad es que simplifica la lectura de llamadas a funciones anidadas (Bache, 2014). Es frecuente encontrar ejemplos donde se usa el operador `%>%`, definido en el paquete `magrittr`, que era la alternativa previa a la introducción del operador *pipe* a partir de la versión 4.1.0 de R.

Otra transformación que se usa a menudo es la de pivotar la matriz de datos, cuyo efecto es el de "alargar" o "ensanchar" la matriz. En el primer caso, se incrementa la cantidad de filas (observaciones) a la vez que se reduce la cantidad de columnas (variables). Para ello se usa la función `pivot_longer(cols, names_to, values_to)` del paquete `tidyr`, donde:

- `cols`: nombres de las columnas a pivotar.
- `names_to`: especifica el nombre de una nueva columna cuyos valores corresponden a los nombres de las columnas a pivotar.
- `values_to`: especifica el nombre de una nueva columna donde se almacenan los valores de las columnas a pivotar.

En el segundo caso se obtiene como resultado una reducción de la cantidad de filas junto al aumento de la cantidad de columnas. Para ello se usa la función `pivot_wider(names_from, values_from)`, también del paquete `tidyr`, donde:

- `names_from`: especifica el nombre de una variable desde la que se obtienen los nombres de las nuevas columnas.
- `values_from`: especifica el nombre de una variable desde donde se obtienen los valores de las nuevas columnas.

Veamos con un ejemplo el efecto de estas dos transformaciones. El script 1.6 comienza por crear una matriz de datos en que se registran los tiempos de ejecución (en milisegundos) para seis instancias de un problema con cuatro algoritmos diferentes. Las columnas de la matriz de datos original corresponden al identificador de la instancia y cada uno de los algoritmos. Así, la matriz de datos original tiene 6 filas y 5 columnas.

A continuación, se crea una nueva matriz de datos, `datos_largos`, que resulta de pivotar la original para "alargarla". Al ejecutar el script 1.6 podemos ver que nuestra nueva matriz de datos tiene solo tres columnas, pero que su cantidad de filas es 24. Si miramos con atención, veremos que ahora tenemos 4 filas por cada instancia, una por cada algoritmo (señalado en la columna `Algoritmo`) con su correspondiente tiempo de ejecución (columna `Tiempo`).

Por último, el script 1.6 crea otro conjunto de datos, `datos_anchos`, a partir de `datos_largos`. Al examinar este nuevo conjunto, se puede apreciar que es idéntico al creado inicialmente.

Script 1.6: modificación del formato de una matriz de datos con el paquete `tidyr`.

```
library(dplyr)
library(tidyr)

# Crear el data frame.
Instancia <- 1:6
Quicksort <- c(23.2, 22.6, 23.4, 23.3, 21.8, 23.9)
Bubblesort <- c(31.6, 29.3, 30.7, 30.8, 29.8, 30.3)
Radixsort <- c(30.1, 28.4, 28.7, 28.3, 29.9, 29.1)
Mergesort <- c(25.0, 25.7, 25.7, 23.7, 25.5, 24.7)
datos <- data.frame(Instancia, Quicksort, Bubblesort, Radixsort, Mergesort)

# Mostrar las primeras filas de la matriz de datos.
cat("Datos originales\n")
print(head(datos))
```

```r
cat("\n")

# Convertir la matriz de datos a formato largo.
datos_largos <- datos |> pivot_longer(c("Quicksort", "Bubblesort",
                                        "Radixsort", "Mergesort"),
                                      names_to = "Algoritmo",
                                      values_to = "Tiempo")

# Mostrar las primeras filas de la matriz de datos largos.
cat("Datos largos\n")
print(head(datos_largos))
cat("\n")

# Convertir la matriz de datos largos a formato ancho.
datos_anchos <- datos_largos |> pivot_wider(names_from = "Algoritmo",
                                            values_from = "Tiempo")

# Mostrar las primeras filas de la matriz de datos largos.
cat("Datos anchos\n")
print(head(datos_anchos))
cat("\n")
```

Habrás notado que para poder usar las funciones de `tidyr` se requiere también el paquete `dplyr`. Una alternativa es cargar únicamente el paquete `tidyverse`, el cual los incluye a ambos (entre otros).

Puedes encontrar descripciones más extensas acerca del uso de la funciones del paquete `tidyverse`, junto con ejemplos más avanzados, en Wickham (2021).

En ocasiones puede ser necesario renombrar las columnas para que nos resulte más fácil comprender a qué variable corresponde. La función `rename()` del paquete `dplyr` nos permite hacer esta operación bastante sencilla. Sus argumentos son una lista de elementos de la forma `nuevo nombre = nombre original`. También podemos cambiar el tipo de una variable. Una conversión que nos será muy útil es de variable numérica a categórica, lo que se logra mediante la función `factor(x, levels, labels, ordered)`, donde:

- `x`: nombre de la variable a convertir.
- `levels`: argumento opcional con los posibles valores de la variable categórica (sus niveles).
- `labels`: argumento opcional con las etiquetas asociadas a cada nivel.
- `ordered`: valor lógico que especifica si la variable es o no ordinal (falso por defecto).

Tomemos el conjunto de datos `mtcars` (incluido en R) para ejemplificar el uso de estas funciones. La tabla 1.3 muestra la descripción de estos datos. El script 1.7 modifica los nombres de las columnas para que sean más representativos y da formato de variable categórica a las variables que así lo requieren, asignando etiquetas adecuadas para cada nivel.

Script 1.7: modificación del conjunto de datos `mtcars` para facilitar su comprensión.

```r
library(dplyr)

# Cargar conjunto de datos y filtrar pesos muy bajos o muy altos.
datos <- mtcars |> filter(wt > 2 & wt < 5)

# Renombrar columnas.
datos <- datos %>% rename(Rendimiento = mpg, Cilindrada = cyl,
                          Desplazamiento = disp, Potencia = hp,
                          Eje = drat, Peso = wt, Cuarto_milla = qsec,
                          Motor = vs, Transmision = am, Cambios = gear,
                          Carburadores = carb)

# Dar formato categórico a las variables Motor y Transmision, renombrando
```

Variable	Descripción
mpg	Rendimiento, en millas / galón (EEUU).
cyl	Número de cilindros.
disp	Desplazamiento, en pulgadas cúbicas.
hp	Potencia, en caballos de fuerza brutos.
drat	Razón del eje trasero.
wt	Peso, en miles de libras.
qsec	Tiempo que tarda en recorrer un cuarto de milla partiendo desde el reposo, en segundos.
vs	Tipo de motor (0: en forma de V, 1: recto).
am	Tipo de transmisión (0: automática, 1: manual).
gear	Número de marchas hacia adelante.
carb	Número de carburadores.

Tabla 1.3: descripción de las variables para el conjunto de datos `mtcars`.

```
# sus niveles.
datos[["Motor"]] <- factor(datos[["Motor"]], levels = c(0, 1),
                           labels = c("V", "Recto"))

datos[["Transmision"]] <- factor(datos[["Transmision"]], levels = c(0, 1),
                                 labels = c("Automático", "Manual"))

# Dar formato ordinal a las variables Cilindrada y Cambios, renombrando
# sus niveles.
datos[["Cilindrada"]] <- factor(datos [["Cilindrada"]], levels = c(4, 6, 8),
                                labels = c("4 cilindros", "6 cilindros",
                                           "8 cilindros"),
                                ordered = TRUE)

datos[["Cambios"]] <- factor(datos[["Cambios"]], levels = c(3, 4, 5),
                             labels = c("3 cambios", "4 cambios", "5 cambios"),
                             ordered = TRUE)

write.csv2(datos , "Mtcars.csv")
```

1.3.5 Fórmulas

Si bien hasta ahora solo tenemos una definición preliminar de lo que es un modelo estadístico, necesitamos conocer herramientas para representarlos. En R se utiliza un objeto llamado **fórmula**, que es necesario manejar bien pues son una parte fundamental del funcionamiento de este lenguaje.

Para entender de manera sencilla qué es una fórmula en R, podemos decir que este objeto permite capturar una **expresión no evaluada** que está asociada a **un ambiente** donde están definidas las variables involucradas en ella. Su sintaxis básica tiene la forma `<variable$_1$>` $\sim$ `<variable$_2$>`, lo que se lee como: en este modelo, los valores de `<variable$_1$>` pueden determinarse a partir de los valores de `<variable$_2$>`. Es decir, las fórmulas representan una relación matemática entre variables. Como se pueden utilizar los valores de una para estimar los valores de la otra, se suele encontrar la forma `<variable dependiente>` $\sim$ `<variable independiente>`. Por ejemplo, podríamos escribir la fórmula `calificación ~ horas.estudio` para representar un modelo en que se puede estimar la calificación que se va a obtener en un examen a partir de las horas de estudio invertidas por cada estudiante.

Podemos refinar un poco más este modelo: `calificación ~ horas.estudio + asignatura`, para indicar que la estimación de la calificación en realidad depende de dos variables: las horas de estudio dedicadas, pero también de la asignatura de que se trate. Intuitivamente podríamos pensar que este modelo podría estimar

mejor que el anterior, puesto que estudiar 5 horas podría ser suficiente para un examen de Español I pero no para uno de Cálculo Integral Avanzado.

Por supuesto podemos refinar más y más este modelo y, por ejemplo, considerar el tipo de asignatura, la institución donde se imparte, el tipo de examen, etc. De hecho la sintaxis de las fórmulas de R permiten incluir y quitar del modelo variables, interacciones entre variables, relaciones de anidación entre variables, relaciones condicionales entre variables, nuevas variables en base a operaciones de otras variables, entre otras cosas. Sin embargo, la mayoría de estos conceptos y modelos están fuera del alcance de este libro.

Notemos que esto no solo aplica a variables numéricas. Por ejemplo, considerando una vez más el conjunto de datos `iris`, podríamos especificar `Species ~ Petal.Length + Petal.Width` para indicar que el valor de la variable `Species`, que es categórica, puede modelarse como una función de las variables `Petal.Length` y `Petal.Width`. Esto puede ser un poco contraintuitivo al comienzo, puesto que el sentido común nos indica que es el largo y ancho de los pétalos de una flor iris los que dependen de su especie. Y así es en realidad, solo que estamos pensando en un modelo que permita realizar la **relación inversa**, es decir, determinar la **clase** de algo a partir de sus **características**. Estudiaremos este tipo de modelos más adelante.

Extenderemos las nociones acerca de la sintaxis y uso de fórmulas a medida que avancemos en nuestro aprendizaje, pero si quieres aprender más puedes consultar Willems (2017).

1.4 EJERCICIOS PROPUESTOS

Considera el siguiente escenario:

> Una encuesta reciente preguntó: "después de la jornada laboral usual, ¿cuántas horas dedica a relajarse o a realizar actividades que disfruta?" a una muestra de 580 chilenas y 575 chilenos. Se encontró que el número promedio de horas era de 1,30 ± 0,30 y 1,95 ± 0,25 para cada grupo, respectivamente.

y responde las siguientes preguntas:

1. ¿Cómo sería una matriz de datos para este estudio? Muestra algunas filas de ella como ejemplos.
2. ¿Cuál podría ser la población objetivo? ¿Qué se entendería por unidad de observación?
3. ¿Qué tipo de variable sería "el número de horas dedicadas a distraerse después de la jornada laboral usual" que respondió cada persona entrevistada? ¿Existe alguna variable categórica? Si es así, ¿de qué tipo? ¿Con qué niveles?
4. ¿Qué dato(s) en el enunciado correspondería(n) a un estadístico? ¿Cuál(es) sería(n) el(los) parámetro(s) en estudio?
5. ¿Puede el estudio establecer que ser mujer chilena ocasiona tener menos horas dedicadas a distraerse después de la jornada laboral usual?

También responde estas otras preguntas:

6. Construye en R una matriz de datos para almacenar las características de una muestra de servidores. Considera a lo menos una variable categórica y una variable numérica.
7. Da un ejemplo de una pregunta de investigación sobre las asignaturas comunes en ingeniería e indica qué variables habría que recolectar para responderla, indicando sus tipos, descripción y rangos.
8. Da un ejemplo de una pregunta de investigación sobre los conciertos realizados en Santiago e indica qué variables habría que recolectar para responderla, indicando sus tipos, descripción y rangos.
9. Da un ejemplo de una pregunta de investigación sobre el estado de la salud mental de estudiantes universitarios e indica qué variables habría que recolectar para responderla, indicando sus tipos, descripción y rangos.
10. Da un ejemplo de dos variables numéricas que están relacionadas positivamente, pero que el aumento en una de ellas no es la causa del aumento en la otra.

11. Da un ejemplo de dos variables numéricas que están relacionadas negativamente, pero que el aumento (o disminución) en una de ellas no es la causa de la disminución (o aumento) en la otra.
12. Investiga para qué sirven y cómo se usan los argumentos `row.names` y `col.names` en las funciones para importar datos desde archivos y la función `data.frame()`.
13. Investiga qué función (o funciones) ofrece R para guardar una matriz de datos en un archivo y úsala(s) para guardar la matriz de datos del ejercicio anterior.
14. Resuelve en R los siguientes ejercicios. Considera para ello el conjunto de datos nativo de R `chickwts`.
    a) ¿Cómo se puede cargar el conjunto de datos en una variable llamada `pollos`?
    b) ¿Cómo se ve la estructura de la matriz de datos almacenada en `pollos`?
15. Muestra ejemplos de las distintas transformaciones que se pueden hacer a una matriz de datos usando para ello conjunto de datos nativo de R `ChickWeight`.

1.5 BIBLIOGRAFÍA DEL CAPÍTULO

Bache, S. (2014). *Introducing magrittr*. Consultado el 7 de abril de 2021, desde https://cran.r-project.org/web/packages/magrittr/vignettes/magrittr.html

Carchedi, N., De Mesmaeker, D., & Vannoorenberghe, L. (s.f.). RDocumentation. Consultado el 2 de abril de 2021, desde https://www.rdocumentation.org/

Dagnino, J. (2014). Tipos de datos y escalas de medida. *Revista Chilena de Anestesia*, *42*(2), 109-111.

Devore, J. L. (2008). *Probabilidad y Estadística para Ingeniería y Ciencias* (7.ª ed.). CENAGE Learning.

Diez, D., Barr, C. D., & Çetinkaya-Rundel, M. (2017). *OpenIntro Statistics* (3.ª ed.). https://www.openintro.org/book/os/.

Freedman, D. A. (2009). *Modelización*. Cambridge University Press.

Joly, F. (1988). *La Cartografía*. Oikos-Tau, S.A. Ediciones.

Kaplan, D. (2009). *Statistical Modeling: A Fresh Approach*. Consultado el 8 de marzo de 2019, desde http://works.bepress.com/daniel_kaplan/38

McCullagh, P. (2002). What Is a Statistical Model? *The Annals of Statistics*, *30*(5), 1225-1267.

Mendez Ramírez, I. (1998). Empirismo, método científico y estadística. *Revista de Geografia Agricola (Mexico)*.

Mendez Ramírez, I. (2012). Método Científico: aspectos epistemológicos y metodológicos para el uso de la Estadística. *SaberEs*, *4*.

Müller, K. (2021). *dplyr*. Consultado el 10 de septiembre de 2021, desde https://dplyr.tidyverse.org/

Posit Software PBC. (2024). RStudio IDE. Consultado el 1 de marzo de 2024, desde https://posit.co/downloads/

Real Academia Española. (2014). *Diccionario de la lengua española* (23.ª ed.). Consultado el 30 de marzo de 2021, desde https://dle.rae.es

Ríos, S. (1995). *Modelización*. Alianza Ediciones.

SAS Institute Inc. (2008). SAS/STAT® 9.2 User's Guide.

The R Foundation. (s.f.). Documentation. Consultado el 2 de abril de 2021, desde https://www.r-project.org/other-docs.html

Wickham, H. (2021). tidyr. Consultado el 10 de septiembre de 2021, desde https://r4ds.had.co.nz/index.html

Wickham, H., & Grolemund, G. (2017). *R for Data Science*. https://r4ds.had.co.nz/index.html.

Willems, K. (2017). *Formulas in R Tutorial*. Consultado el 11 de septiembre de 2021, desde https://dplyr.tidyverse.org/

CAPÍTULO 2. EXPLORACIÓN DE DATOS

Siempre es bueno que nos familiaricemos con los datos y algunas de sus características antes de empezar a trabajar con ellos. Esto nos ayuda a decidir qué herramientas son las más adecuadas para dar respuesta a las preguntas que queramos responder. En este capítulo revisaremos las principales estadísticas descriptivas que nos ayudarán a resumir los datos para entenderlos mejor, así como diversos tipos de gráficos que nos permitirán representar los datos de modo que podamos comprenderlos de forma visual. Para ello, tomamos como base los conceptos expuestos en Diez et al. (2017, pp. 26-50), Field et al. (2012, pp. 19-27) y STDHA (s.f.), fuentes que puedes consultar si deseas saber más acerca de estos temas.

Para muchos de los ejemplos de este capítulo usaremos el conjunto de datos `mtcars` con las modificaciones realizadas en el script 1.7, cuyo diccionario de datos se muestra en la tabla 2.1.

Variable	Descripción
Rendimiento	Rendimiento, en millas / galón (EEUU).
Cilindrada	Número de cilindros (4 cilindros, 6 cilindros, 8 cilindros).
Desplazamiento	Desplazamiento, en pulgadas cúbicas.
Potencia	Potencia, en caballos de fuerza brutos.
Eje	Razón del eje trasero.
Peso	Peso, en miles de libras.
Cuarto_milla	Tiempo que tarda en recorrer un cuarto de milla partiendo desde el reposo, en segundos.
Motor	Tipo de motor (V, Recto).
Transmision	Tipo de transmisión (Automático, Manual).
Cambios	Número de cambios hacia adelante (3 cambios, 4 cambios, 5 cambios).
Carburadores	Número de carburadores.

Tabla 2.1: descripción de las variables para el conjunto de datos `mtcars`.

2.1 ESTADÍSTICAS DESCRIPTIVAS

Las estadísticas descriptivas son medidas que nos permiten sintetizar y, como su nombre lo indica, describir los datos. Estas pueden aplicarse tanto a una muestra como a una población. Cuando una de estas medidas se aplica a la muestra, corresponde a un **estimador puntual** de la misma medida para la población. Al ser una estimación, no es exacta, aunque la precisión tiende a aumentar mientras mayor sea el tamaño de la muestra, y puede varíar de muestra en muestra.

Un concepto importante a tener en cuenta es la noción de **distribución**. En este capítulo se considera la **distribución de frecuencia**, que representa cuántas veces aparece cada valor para una variable en un conjunto de datos.

2.1.1 Estadísticas descriptivas para datos numéricos

Una de las estadísticas descriptivas más empleadas es la **media**, conocida en otros contextos como media aritmética o promedio. Denotamos la **media muestral** por $\overline{x}$, donde x corresponde al nombre de la variable, mientras que para la **media poblacional** empleamos la notación μ_x. Esta medida se calcula como muestra la ecuación 2.1, donde x_i son los n valores observados de la variable. Podemos entender la media como el punto de equilibrio de la distribución (Diez et al., 2017, p. 28). Así, la media corresponde a una **medida de tendencia central**.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

El script 2.1 muestra cómo usar la función `mean()` de R para calcular la media de diversas variables del conjunto de datos `mtcars`[1]. Como primer ejemplo, se calcula la media de la variable `Rendimiento`. A continuación se muestra cómo realizar esta operación para dos variables, señaladas por el índice de sus respectivas columnas. Luego, de manera similar, se calculan las medias para cuatro columnas consecutivas de la matriz de datos. En estos dos casos hacemos uso de la función `sapply()`, que permite aplicar una misma función (cualquiera) para múltiples columnas.

Script 2.1: uso de las funciones `mean()` y `sapply()`.

```
# Cargar conjunto de datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Calcular la media para la variable Rendimiento.
media <- mean(datos[["Rendimiento"]])
cat("Rendimiento medio:", media, "\n\n")

# Calcular la media para la tercera y quinta columnas
# (variables Desplazamiento y Eje).
cat("Medias\n")
print(sapply(datos[c(3, 5)], mean))
cat("\n")

# Calcular la media para las columnas 3 a 6
# (variables Desplazamiento, Potencia, Eje y Peso).
cat("Medias\n")
print(sapply(datos[3:6], mean))
cat("\n")

# Calcular la media para la variable Rendimiento omitiendo valores faltantes.
print(mean(datos[["Rendimiento"]], na.rm = TRUE))
```

La función `mean()` devuelve `NA` (*not available*, es decir, no disponible) si existen valores faltantes en los datos de entrada. Para prevenir este error, se puede proporcionar un argumento adicional que descarte los valores faltantes, como muestra la última línea del script 2.1.

Una medida de tendencia central alternativa a la media es la **mediana**, que es, simplemente, el valor central de los valores previamente ordenados. Cuando no existe un valor central, vale decir, cuando el tamaño de la muestra es par, la mediana está dada por el promedio simple de los dos valores centrales. En R, la mediana se calcula con la función `mkdfunc()`.

La **moda** es, simplemente, el valor más frecuente en el conjunto de datos. No obstante, tiene el problema de que puede haber múltiples modas. Dependiendo de la cantidad de modas, se habla de distribuciones **unimodales**, **bimodales** y **multimodales**.

Si bien R no cuenta con una función nativa para encontrar la moda, el paquete `modeest` ofrece la función `mfv()` que entrega el valor más frecuente de una variable. En caso de que dos (o más) valores sean los más frecuentes con igual cantidad de observaciones, los entrega todos en forma de vector.

Las medidas que hemos estudiado hasta ahora buscan describir el centro del conjunto de datos. No obstante, también es importante conocer su **variabilidad** o dispersión, pues así se puede saber qué tan semejantes (o diferentes) son las observaciones entre sí. Por ejemplo, decir que el ingreso mensual medio del país es $500 mil es incompleto: ¿será que la mayoría de los habitantes tiene un ingreso mensual entre $400 y $600 mil o

[1] Todas las demás funciones de R mencionadas en esta sección para las que no se proporcione un script se usan del mismo modo que `mean()`.

hay personas con ingresos de $20 mil y otros de $1 millón mensuales? Estas son situaciones muy distintas que quedan encubiertas si solo se conoce una medida de tendencia central.

La variabilidad suele calcularse en base a la **desviación** de las observaciones, que se entiende como la distancia entre una observación y la media del conjunto de datos. Las dos principales medidas de dispersión son la **varianza** y la **desviación estándar**, siendo la primera el cuadrado de la segunda.

La varianza muestral se calcula como muestra la ecuación 2.2, donde x_i son los valores de cada una de las n observaciones. Puede verse que su valor se basa en los cuadrados de las desviaciones, ya que, por una parte, los valores grandes se incrementan más significativamente y, por otra, se opera solo con valores positivos, pues la dirección de la distancia no es de interés. Es más, la suma directa de las desviaciones tiende a cero puesto que, en general, aproximadamente la mitad de las observaciones estarán bajo la media (con desviación negativa) mientras que las restantes estarán sobre ella (con desviación positiva).

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{2.2}$$

La desviación estándar de la muestra se define como la raíz cuadrada de la varianza (2.3), medida que resulta de gran utilidad cuando se necesita saber cuán cercanos son los datos a la media, ya que se encuentra en la misma escala que la variable estudiada.

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{2.3}$$

Las funciones de R para calcular la varianza y la desviación estándar son, respectivamente, `var()` y `sd()`.

Al igual que en el caso de la media, podemos usar las fórmulas anteriores para obtener estimaciones puntuales de la varianza y la desviación estándar de la población, denotadas por σ^2 y σ, respectivamente. Es importante considerar que, si bien la media y la desviación estándar permiten conocer el centro y la dispersión del conjunto de datos, respectivamente, la forma que exhiben distribuciones con parámetros similares puede variar considerablemente, como ilustra la figura 2.1.

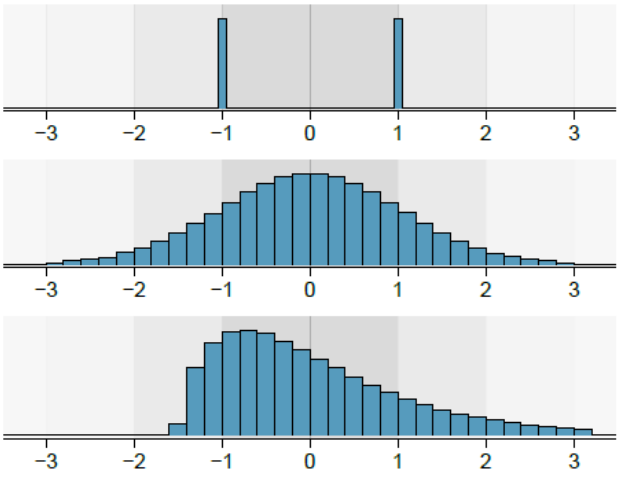


Figura 2.1: tres distribuciones de población muy distintas con media $\mu = 0$ y desviación estándar $\sigma = 1$. Fuente: Diez et al. (2017, p. 34).

Aunque menos empleado, el **rango** muestra los valores extremos, es decir, el mínimo y el máximo, de una variable. R ofrece la función `range()` para obtener ambos valores, además de `min()` y `max()` para obtenerlos por separado.

En párrafos anteriores vimos que la mediana es el valor central (o el promedio de los dos valores centrales) del conjunto de datos ordenado, ya sea una población o una muestra. Esto significa, entonces, que esta medida divide el conjunto de datos en dos mitades con igual cantidad de elementos. De manera similar, es posible dividir el conjunto de datos en segmentos más pequeños, por ejemplo en 4, 10 o 100 partes con igual cantidad de elementos. Cada fragmento del conjunto de datos dividido de esta forma recibe el nombre de **cuantil**. Algunas subdivisiones de uso frecuente reciben nombres especiales:

- **Percentiles**: dividen el conjunto de datos en 100 subconjuntos de igual tamaño.
- **Deciles**: dividen el conjunto de datos en 10 subconjuntos de igual tamaño.
- **Quintiles**: dividen el conjunto de datos en 5 subconjuntos de igual tamaño.
- **Cuartiles**: dividen el conjunto de datos en 4 subconjuntos de igual tamaño.

Los cuantiles (al igual que las otras subdivisiones antes mencionadas) se nombran de forma ascendente según el sentido de crecimiento del conjunto de datos. Así, el percentil 1 contiene a los valores más pequeños, mientras que el percentil 100, a los más grandes. Cabe destacar que la mediana corresponde al percentil 50 o al cuartil 2, y que nombrar al decil 3 es equivalente al percentil 30.

R proporciona la función `quantile()` para calcular cuantiles, que por defecto calcula los cuartiles, aunque su uso puede generalizarse mediante el parámetro adicional `probs`, como muestra el script 2.2. La función `seq()` genera una secuencia de números equiespaciados, y recibe como argumentos el inicio, el término y el incremento de la secuencia.

Script 2.2: cálculo de cuantiles con la función `quantile()`.

```
# Cargar conjunto de datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Cálculo de percentiles para la variable Rendimiento.
cat("Cuartiles:\n")
print(quantile(datos[["Rendimiento"]]))
cat("\n")

cat("Quintiles:\n")
print(quantile(datos[["Rendimiento"]], seq(0, 1, 0.2)))
cat("\n")

cat("Deciles:\n")
print(quantile(datos[["Rendimiento"]], seq(0, 1, 0.1)))
cat("\n")

cat("Percentiles:\n")
print(quantile(datos[["Rendimiento"]], seq(0, 1, 0.01)))
```

Ahora que conocemos los cuartiles, podemos introducir una nueva medida de variabilidad que usaremos a menudo, llamada **rango intercuartil** o IQR (por su sigla en inglés), dada por la ecuación 2.4, donde Q_1 y Q_3 corresponden a los cuartiles 1 y 3, respectivamente. Al igual que la varianza y la desviación estándar, mientras más disperso sea el conjunto de datos, mayor será el valor del IQR. En R, la función que calcula este estimador es `IQR()`.

$$IQR = Q_3 - Q_1 \tag{2.4}$$

Muchas veces los conjuntos de datos contienen lo que se conoce como **valores atípicos** u ***outliers***. Estos corresponden a observaciones que parecen estar fuera de rango o ser muy extremos con respecto al resto de los datos. Medidas como la media o la desviación estándar son muy sensibles a los valores atípicos y pierden capacidad descriptiva ante la presencia de este tipo de observaciones. Para reducir el efecto de los valores extremos muchas veces necesitaremos medidas **robustas**, que son aquellas que proporcionan una estimación

confiable aún ante la presencia de valores atípicos. En este escenario, la mediana y el IQR resultan ser medidas de tendencia central y de dispersión, respectivamente, más robustas.

Nos encontraremos frecuentemente con la necesidad de calcular varias medidas de tendencia central y de dispersión descritas en el apartado anterior. Por esta razón, R, y algunos de sus paquetes, ofrecen algunas funciones que calculan varios de estos estadísticos con una sola llamada. Tal es el caso de la función nativa `summary()`, que entrega la media, la mediana, el primer y el tercer cuartil, el mínimo y el máximo. Otra función que nos puede ser de mucha ayuda es `summarise()`, del paquete `dplyr`. Con ella podemos calcular varias de las medidas en una sola llamada, como muestra el script 2.3.

Script 2.3: uso de la función `summarise()` del paquete `dplyr`.

```
library(dplyr)

datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Cálculo de varias medidas para la variable Potencia.
medidas_potencia <- datos %>% summarise(Media = mean(Potencia),
                                        Mediana = median(Potencia),
                                        Varianza = var(Potencia),
                                        IQR = IQR(Potencia))

print(medidas_potencia)
cat("\n")

# Cálculo de la media y la desviación estándar para las variables Peso y
# Cuarto_milla.
medidas_varias <- datos %>% summarise(Media_P = mean(Peso),
                                      Media_C = median(Cuarto_milla),
                                      SD_P = sd(Peso),
                                      SD_C = sd(Cuarto_milla))

print(medidas_varias)
cat("\n")
```

2.1.2 Estadísticas descriptivas para datos categóricos

Cuando queremos trabajar con datos categóricos, medidas como la media o la desviación estándar carecen de sentido. En consecuencia, necesitamos otros estadísticos para resumir el conjunto de datos.

Como primer estadístico para variables categóricas podemos mencionar la **frecuencia**, que corresponde a la cantidad de veces que podemos encontrar cada nivel de la variable en los datos. Otro estadístico importante corresponde a la **proporción**, que corresponde a la frecuencia relativa. En otras palabras, la proporción corresponde a frecuencia de un nivel de la variable dividida por la cantidad total de observaciones.

La mejor alternativa para este tipo de datos es la **tabla de contingencia**, también llamada **matriz de confusión** o **tabla de frecuencias**, donde cada fila representa la cantidad de veces en que ocurre una combinación de variables. También es posible usar porcentajes o proporciones en lugar de la cantidad de ocurrencia, en cuyo caso se habla de una **tabla de frecuencias relativas**. La tabla 2.2 muestra la tabla de contingencia (de frecuencias) para la variable `Cambios` del ejemplo. Se puede observar, entre otras cosas, que el conjunto de datos contiene una muestra de 32 automóviles y que 15 de ellos tienen tres cambios.

3 cambios	4 cambios	5 cambios	Total
12	9	4	25

Tabla 2.2: tabla de contingencia para la cantidad de cambios de los automóviles.

Desde luego, podemos construir tablas de contingencia de manera bastante sencilla en R. El script 2.4 muestra dos formas de obtener la tabla 2.2. La primera es la función `table()` y la segunda, la función `xtabs()`. El funcionamiento de ambas es equivalente, aunque `xtabs()` muestra el nombre de la variable tabulada al imprimir los resultados y `table()` no lo hace. Las tablas entregadas por estas funciones no incluyen los totales por filas, pero la función `marginSums()` permite calcularlos y mostrarlos como un vector. A su vez, la función `addmargins()` permite calcular dichos totales e incorporarlos a la tabla. Para terminar, el las últimas sentencias del script 2.4 ilustran la manera de obtener las tablas de frecuencias relativas con proporciones y porcentajes, respectivamente.

Podemos ver que las llamadas a `table()` y a `xtabs()` son algo diferentes. La primera recibe como argumento la columna de la matriz de datos, es decir, un vector con los datos a tabular, mientras que la segunda recibe una fórmula en que no existe una variable dependiente y la variable categórica es la independiente.

Script 2.4: tabla de contingencia para la variable `Cambios`.

```r
# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear tabla de contingencia para la variable Cambios.
contingencia <- table(datos[["Cambios"]])
cat("Tabla de contingencia generada con table():\n")
print(contingencia)
cat("\n")

# Otra forma de crear la misma tabla.
contingencia <- xtabs(~ Cambios, data = datos)
cat("Tabla de contingencia generada con xtabs():\n")
print(contingencia)
cat("\n")

# Calcular totales por fila y mostrarlos por separado.
totales <- marginSums(contingencia)
cat("Totales por fila:\n")
print(totales)
cat("\n")

# Calcular totales por fila y agregarlos a la tabla.
con_totales <- addmargins(contingencia, 1)
cat("Tabla de contingencia con totales por fila:\n")
print(con_totales)
cat("\n")

# Convertir a tabla de proporciones
proporciones <- prop.table(contingencia)
proporciones <- addmargins(proporciones, 1)
cat("Tabla de contingencia con proporciones:\n")
print(proporciones)
cat("\n")

# Convertir a tabla de porcentajes con 2 decimales.
porcentajes <- round(prop.table(contingencia), 4) * 100
porcentajes <- addmargins(porcentajes)
cat("Tabla de contingencia con porcentajes:\n")
print(porcentajes)
cat("\n")
```

También podemos construir matrices de confusión para dos variables categóricas, como muestra la tabla 2.3 para las variables `Cambios` y `Transmisión`.

En ocasiones resulta útil determinar las proporciones por fila o por columna, que podemos obtener dividiendo

		Cambios			
		3 cambios	4 cambios	5 cambios	Total
Transmisión	Automático	12	4	0	16
	Manual	0	5	4	9
	Total	12	9	4	25

Tabla 2.3: tabla de contingencia para las variables `Cambios` y `Transmisión`.

el valor de una celda de la matriz por el total de su fila o columna, según corresponda. Así, el total de cada fila (o columna) es igual a 1. Puesto que las proporciones por fila y por columna no son equivalentes, debemos ser cuidadosos al escoger la más adecuada en cada caso. Las tablas 2.4 a 2.6 muestran las proporciones por fila, por columna y generales para la matriz de confusión de la tabla 2.3. La construcción en R de la tabla de contingencia y las tablas de proporciones para dos variables se muestra en el script 2.5.

		Cambios			
		3 cambios	4 cambios	5 cambios	Total
Transmisión	Automático	0,7500000	0,2500000	0,0000000	1,0000000
	Manual	0,0000000	0,5555556	0,4444444	1,0000000

Tabla 2.4: tabla de proporciones con totales por fila para la tabla 2.3.

		Cambios		
		3 cambios	4 cambios	5 cambios
Transmisión	Automático	1,0000000	0,4444444	0,0000000
	Manual	0,0000000	0,5555556	1,0000000
	Total	1,0000000	1,0000000	1,0000000

Tabla 2.5: tabla de proporciones con totales por columna para la tabla 2.3.

Script 2.5: tablas de contingencia y proporciones para dos variables.

```r
# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear tabla de contingencia para las variables Transmisión y Cambios.
contingencia <- table(datos[["Transmision"]], datos[["Cambios"]])
cat("Tabla de contingencia generada con table():\n")
print(contingencia)
cat("\n")

# Otra forma de crear la misma tabla.
contingencia <- xtabs(~ Transmision + Cambios, data = datos)
cat("Tabla de contingencia generada con xtabs():\n")
print(contingencia)
cat("\n")

# Proporciones con totales por fila.
proporciones_fila <- prop.table(contingencia, margin=1)
proporciones_fila <- addmargins(proporciones_fila, margin=2)
cat("Tabla de contingencia con proporciones totales por fila:\n")
print(proporciones_fila)
cat("\n")

# Proporciones con totales por columna.
proporciones_columna <- prop.table(contingencia, margin=2)
proporciones_columna <- addmargins(proporciones_columna, margin=1)
cat("Tabla de contingencia con proporciones totales por columna:\n")
```

		Cambios			
		3 cambios	4 cambios	5 cambios	Total
Transmision	Automático	0,48	0,16	0,00	0,64
	Manual	0,00	0,20	0,16	0,36
	Total	0,48	0,36	0,16	1,00

Tabla 2.6: tabla de proporciones con totales por fila y columna para la tabla 2.3.

```r
print(proporciones_columna)
cat("\n")

# Proporciones con totales.
proporciones <- prop.table(contingencia)
proporciones <- addmargins(proporciones)
cat("Tabla de contingencia con proporciones totales:\n")
print(proporciones)
cat("\n")
```

Aunque no se usa con frecuencia, podríamos necesitar una matriz de confusión para más de dos variables. Veamos ahora un ejemplo con tres variables: `Motor`, `Cambios` y `Transmisión`. Para ello, tomamos una de las variables (en este caso, `Motor`) y creamos una subtabla por cada uno de sus niveles. Cada subtabla muestra las frecuencias para la combinación de las dos variables restantes cuando `Motor` tiene el nivel correspondiente, como muestra la tabla 2.7. En R, podemos obtener estas tablas como muestra el script 2.6. Desde luego, esta misma idea, y con algo de imaginación, puede extenderse para cuatro o más variables categóricas.

Motor = Recto

		Cambios		
		3 cambios	4 cambios	5 cambios
Transmision	Automático	3	4	0
	Manual	0	3	0

Motor = V

		Cambios		
		3 cambios	4 cambios	5 cambios
Transmision	Automático	9	0	0
	Manual	0	2	4

Tabla 2.7: tabla de contingencia para tres variables.

Script 2.6: matriz de confusión para tres variables.

```r
# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Convertir la variable Cambios en categórica.
datos[["Cambios"]] <- factor(datos[["Cambios"]])

# Crear tabla de contingencia para las variables Transmision,
# Cambios y Motor.
contingencia <- ftable(datos[["Transmision"]], datos[["Cambios"]],
                       datos[["Motor"]])

cat("Tabla de contingencia generada con ftable():\n")
print(contingencia)
cat("\n")

# Otra forma de crear la misma tabla.
```

```
xtabs(~ Cambios + Transmision + Motor, data = datos)
cat("Tabla de contingencia generada con xtabs():\n")
print(contingencia)
cat("\n")
```

2.1.3 Trabajando con datos agrupados

A menudo nos veremos en la necesidad de obtener estadísticas descriptivas de una variable separando las observaciones en grupos de acuerdo a una variable categórica. Para ello, el paquete `dplyr` ofrece la función `group_by()`, que podemos usar en conjunto con `summarise()`, como muestra el script 2.7. En dicho script, primero se agrupan las observaciones de acuerdo a la variable `Cambios`, y luego se efectúa una llamada a `summarise()` donde el primer argumento cuenta la cantidad de observaciones en el grupo actual y los argumentos restantes (que pueden ser tantos como se desee) corresponden a diferentes estadísticas descriptivas.

Script 2.7: estadísticas descriptivas para datos agrupados.

```
library(dplyr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

resumen <- group_by(datos, Cambios) |>
  summarise(count = n(), mean(Rendimiento), median(Rendimiento),
            sd(Rendimiento), IQR(Rendimiento), mean(Potencia))

print(resumen)
```

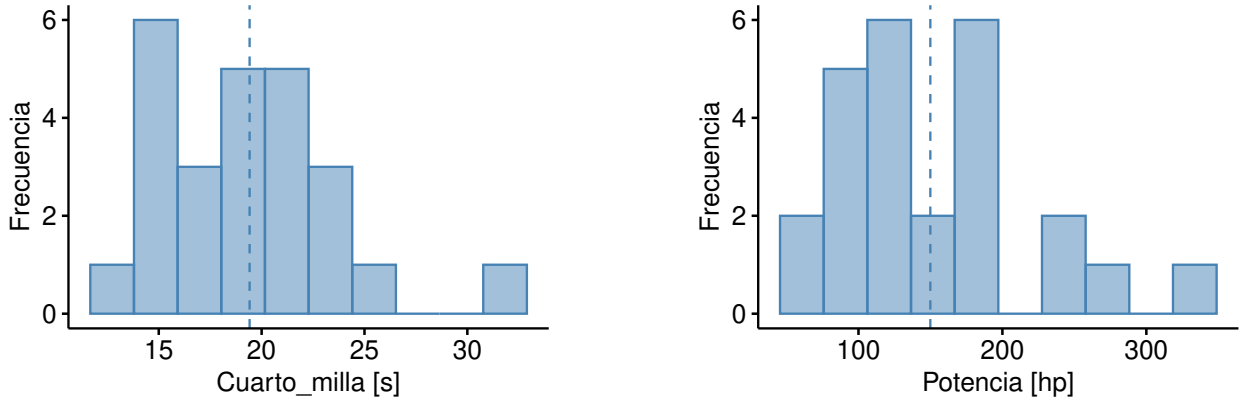
2.2 REPRESENTACIÓN GRÁFICA DE DATOS

En esta sección revisaremos diversos tipos de gráficos que resultan útiles al momento de estudiar un conjunto de datos disponibles, considerando su definición, su utilidad y cómo se construyen en R. Para crear gráficos en R usaremos funciones del paquete `ggpubr`. Algunos de los principales parámetros que usaremos para crear y editar gráficos con este paquete son:

- `data`: una matriz de datos (data frame).
- `x`: string con el nombre de la variable asociada al eje x.
- `y`: string que identifica los valores a graficar, usualmente el nombre de una variable.
- `color`: color de delineado.
- `fill`: color de relleno.
- `palette`: paleta de colores que usualmente se usa cuando existen múltiples grupos.
- `linetype`: tipo de línea a emplear.
- `add`: permite agregar elementos adicionales al gráfico, como barras de error o la media, entre otros.
- `title`: título del gráfico.
- `xlab`: rótulo del eje x. Puede ocultarse usando `xlab = FALSE`.
- `ylab`: rótulo del eje y. Puede ocultarse usando `ylab = FALSE`.

2.2.1 Una variable numérica

El **histograma** resulta muy útil si queremos representar una única variable numérica y la muestra es grande. Podemos decir que este gráfico muestra una aproximación a la **densidad** (o distribución de frecuencias) para la variable, para lo que tenemos que dividir el rango de valores posibles en intervalos (generalmente iguales) y luego contar la cantidad de observaciones en cada intervalo. Para construir el gráfico, creamos una barra por cada intervalo, cuya altura (o longitud) es proporcional a la cantidad de observaciones en el intervalo representado. La figura 2.2 muestra histogramas creados con el script 2.8.



(a) Distribución cercana a la simétrica.

(b) Distribución desviada a la izquierda.

Figura 2.2: dos histogramas.

Script 2.8: histogramas para las variables `Rendimiento` y `Potencia`.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("C:/Inferencia/Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Histograma para la variable Rendimiento.
g1 <- gghistogram(datos,
                  x = "Rendimiento",
                  bins = 10,
                  add = "mean",
                  xlab = "Rendimiento [Millas/galón]",
                  ylab = "Frecuencia",
                  color = "blue",
                  fill = "blue")

print(g1)

# Histograma para la variable Potencia.
g2 <- gghistogram(datos,
                  x = "Potencia",
                  bins = 10,
                  add = "mean",
                  xlab = "Potencia [hp]",
                  ylab = "Frecuencia",
                  color = "red",
                  fill = "yellow")

print(g2)
```

A medida que avancemos en este libro, veremos que es muy importante conocer la **distribución de frecuencias** de una variable. Al observar la figura 2.2b, podemos ver que la frecuencia es mayor para potencias más bajas, pues las barras de la izquierda del gráfico son, en general, algo más altas que las de la derecha. Podría decirse que las observaciones se concentran a la izquierda y que hay una cola que se prolonga hacia la derecha. Cuando esto ocurre, decimos que la la distribución está **desviada a la izquierda**, o que hay **asimetría negativa**. Análogamente, podría darse que la distribución estuviese desviada a la derecha o, equivalentemente, que presenta asimetría positiva. En el caso de la figura 2.2a, el histograma es más **simétrico**, pues las observaciones se aglomeran hacia el centro y hay colas tanto a la izquierda como a la derecha. Para ilustrar mejor la idea de la simetría, podemos revisar una vez más la figura 2.1, donde la población central es perfectamente simétrica y la inferior presenta asimetría positiva.

Otra ventaja de los histogramas es que permiten identificar modas de una variable, las cuales corresponden a barras que sean más prominentes que las de su entorno. Ambos ejemplos de la figura 2.2 son bimodales, pues tienen dos modas claramente identificables. Si bien es cierto que en ambos casos hay un único valor más frecuente (moda), podemos apreciar que existen dos "cumbres" o máximos locales.

Otro gráfico que usaremos a menudo es el de **gráfico de caja**. Es muy útil, pues su construcción considera 5 estadísticos para representar el conjunto de datos y además facilita la identificación de datos atípicos. La figura 2.3 muestra este gráfico para la variable `Potencia`, el cual fue creado con el script 2.9.

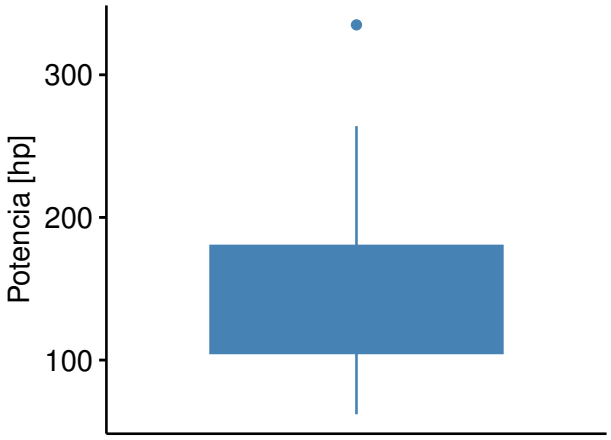


Figura 2.3: gráfico de caja para la variable `Potencia`.

Los extremos inferior y superior del rectángulo o caja de la figura 2.3 corresponden, respectivamente, al primer y al tercer cuartil, mientras que la línea horizontal al interior de la caja denota la mediana. Así, la caja engloba el 50 % central de los datos, y su altura corresponde al rango intercuartil. Las barras que se extienden por sobre y por debajo de la caja, llamadas "bigotes", capturan aquellos datos fuera de la caja central y que estén situados a no más de 1,5 veces el IQR. Cualquier observación que esté más allá de la caja y los bigotes se representa como un punto, ya que, bajo este criterio, se considera una observación atípica.

Script 2.9: gráfico de caja para la variable `Potencia`.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

g <- ggboxplot(datos[["Potencia"]],
               color = "steelblue",
               fill = "steelblue",
               ylab = "Potencia [hp]")

g <- g + rremove("x.ticks")
```

```
g <- g + rremove("x.text")
g <- g + rremove("x.title")

print(g)
```

2.2.2 Una variable categórica

Si queremos representar una única variable categórica, lo más adecuado es usar un **gráfico de barras**, pues cada barra es tan larga como la proporción de valores presentes en cada nivel de la variable. La figura 2.4 muestra el gráfico de barras correspondiente a la tabla 2.2, elaborado mediante el script 2.10.

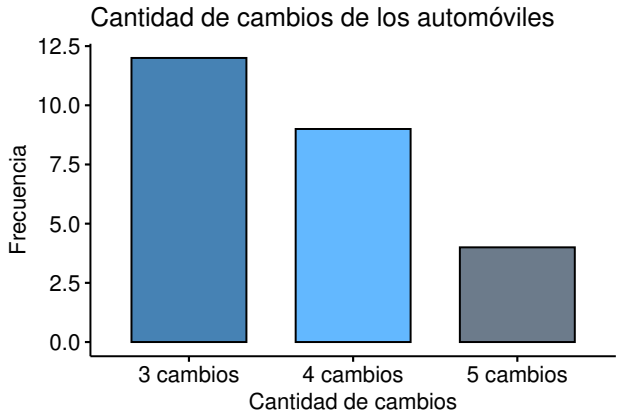


Figura 2.4: gráfico de barras para la variable `Cambios`.

Script 2.10: gráfico de barras para la variable `Cambios`.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear la tabla de frecuencias para la variable Cambios y convertirla a
# data frame.
contingencia <- as.data.frame(xtabs(~ Cambios, data = datos))

# Crear el gráfico de barras.
g <- ggbarplot(contingencia,
               x = "Cambios",
               y = "Freq",
               fill = "Cambios",
               palette = c("steelblue", "steelblue1", "slategray4"),
               title = "Cantidad de cambios de los automóviles",
               xlab = "Cantidad de cambios",
               ylab = "Frecuencia")
g <- ggpar(g, legend = "none")

print(g)
```

Otra alternativa para representar una única variable categórica es el **gráfico de torta**, que se presenta en la figura 2.5 y se construye en R como muestra el script 2.11.

Script 2.11: gráfico de torta para la variable `Cambios`.

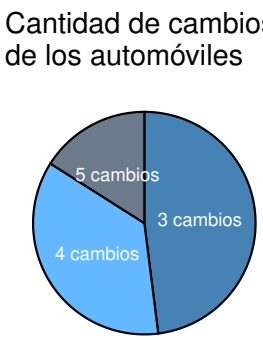


Figura 2.5: gráfico de torta para la variable `Cambios`.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear la tabla de frecuencias y convertirla a data frame.
contingencia <- as.data.frame(xtabs(~ Cambios, data = datos))

# Crear gráfico de torta.
g <- ggpie(contingencia,
           x = "Freq",
           label = "Cambios",
           lab.font= c(3, "plain", "white"),
           fill = c("steelblue", "steelblue1", "slategray4"),
           title = "Cantidad de cambios\nde los automóviles",
           lab.pos = "in")

print(g)
```

2.2.3 Dos variables numéricas

Los **gráficos de dispersión** son adecuados en este caso. Se caracterizan porque muestran información caso a caso, ya que cada punto del gráfico corresponde a una observación. Por ejemplo, el gráfico de la figura 2.6, creado mediante el script 2.12, muestra este tipo de gráfico para las variables `Rendimiento` y `Peso`.

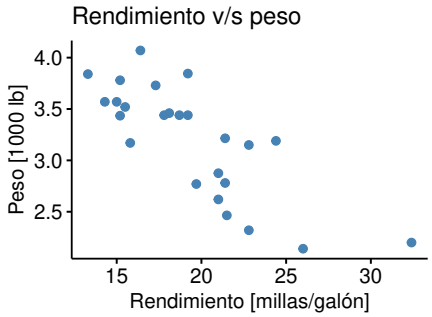


Figura 2.6: gráfico de dispersión para las variables `Rendimiento` y `Peso`.

Script 2.12: gráfico de dispersión para las variables `Rendimiento` y `Peso`.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear gráfico de dispersión.
g <- ggscatter(datos,
               x = "Rendimiento",
               y = "Peso",
               color = "steelblue",
               title = "Rendimiento v/s peso",
               xlab = "Rendimiento [millas/galón]",
               ylab = "Peso [1000 lb]")

print(g)
```

Los gráficos de dispersión también son muy útiles para identificar si dos (o más) variables están relacionadas. La figura 2.7 (creada mediante el script 2.13) muestra tres gráficos de dispersión diferentes: en el de la izquierda, se aprecia que las variables `Peso` y `Cuarto_milla` son independientes, pues no hay una tendencia definida en la organización de los puntos. En el gráfico del centro, en cambio, podemos ver que la potencia tiende a aumentar a medida que también lo hace el peso, por lo que ambas variables están positivamente asociadas. Por último, el gráfico de la derecha nos muestra que las variables `Peso` y `Rendimiento` presentan asociación negativa, puesto que a medida que la primera aumenta, la segunda disminuye.

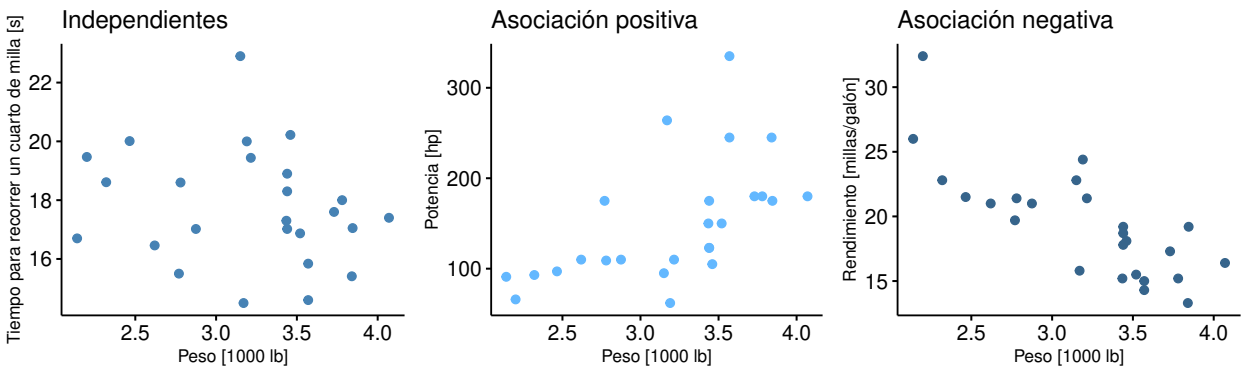


Figura 2.7: gráficos de dispersión con diferentes tipos de asociación entre las variables.

Script 2.13: gráficos de dispersión con diferentes tipos de asociación entre las variables.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Gráfico para variables independientes.
g1 <- ggscatter(datos,
                x = "Peso",
                y = "Cuarto_milla",
                color = "steelblue",
                title = "Independientes",
                xlab = "Peso [1000 lb]",
                ylab = "Tiempo para recorrer un cuarto de milla [s]")
g1 <- ggpar(g1, font.x = c(10, "plain", "black"),
                font.y = c(10, "plain", "black"))
```

```r

# Gráfico para variables con asociación positiva.
g2 <- ggscatter(datos,
                x = "Peso",
                y = "Potencia",
                color = "steelblue1",
                title = "Asociación positiva",
                xlab = "Peso [1000 lb]",
                ylab = "Potencia [hp]")
g2 <- ggpar(g2, font.x = c(10, "plain", "black"),
                font.y = c(10, "plain", "black"))

# Gráfico para variables con asociación negativa.
g3 <- ggscatter(datos,
                x = "Peso",
                y = "Rendimiento",
                color = "steelblue4",
                title = "Asociación negativa",
                xlab = "Peso [1000 lb]",
                ylab = "Rendimiento [millas/galón]")
g3 <- ggpar(g3, font.x = c(10, "plain", "black"),
                font.y = c(10, "plain", "black"))

# Crear figura con tres gráficos.
g  <- ggarrange(g1 ,g2 ,g3, ncol = 3, nrow = 1, common.legend = TRUE)

print(g)
```

2.2.4 Dos variables categóricas

Similares al gráfico de barras para una variable categórica, los **gráficos de barras apiladas, agrupadas y estandarizadas** permiten visualizar la matriz de confusión entre dos variables y encontrar posibles relaciones entre ellas. La figura 2.8, creada con el script 2.14, ejemplifica esta familia de gráficos usando para ello las variables `Cambios` y `Motor`.

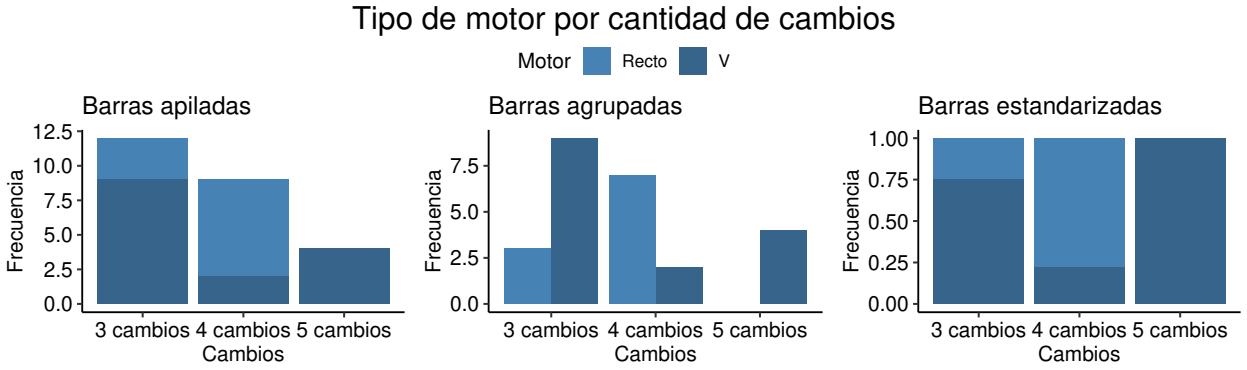


Figura 2.8: gráficos de barras para las variables `Cambios` y `Motor`.

El **gráfico de barras apiladas**, a la izquierda en la figura 2.8, muestra tres barras cuya altura corresponde a la frecuencia de la cantidad de cambios, al igual que en la figura 2.4, pero ahora cada barra está subdividida en secciones de distinto color para cada tipo de motor. La altura de cada sección está dada por la frecuencia del tipo de motor para la cantidad de cambios representada en la barra.

Similar al anterior, el gráfico de la derecha en la figura 2.8, que corresponde a un **gráfico de barras estandarizadas**, muestra barras de igual altura para cada nivel de la variable `Cambios` representando claramente

diferencias en la proporción de cada tipo de motor por cada nivel. Se puede apreciar que los automóviles con 3 y 5 cambios tienen mayoritariamente motores en forma de V, mientras que el uso de motores rectos se da principalmente en automóviles de 4 cambios.

El **gráfico de barras agrupadas**, al centro en la figura 2.8, es equivalente al de la izquierda, pero en lugar de dividir una barra en segmentos, muestra barras contiguas para cada tipo de motor.

Script 2.14: gráficos de barras para las variables `Cambios` y `Motor`.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear tabla de contingencia para las variables Motor y Cambios,
# y guardarla como data frame.
tabla <- xtabs(~ Motor + Cambios, data = datos)
contingencia <- as.data.frame(tabla)

# Crear gráfico de barras segmentadas.
g1 <- ggplot(contingencia, aes(fill = Motor, y = Freq, x = Cambios))
g1 <- g1 + geom_bar(position = "stack", stat = "identity")
g1 <- g1 + scale_fill_manual(values=c("steelblue", "steelblue4"))
g1 <- g1 + labs(y = "Frecuencia") + ggtitle("Barras apiladas")
g1 <- g1 + theme_pubr()

# Crear gráfico de barras agrupadas.
g2 <- ggplot(contingencia, aes(fill = Motor, y = Freq, x = Cambios))
g2 <- g2 + geom_bar(position = "dodge", stat = "identity")
g2 <- g2 + scale_fill_manual(values=c("steelblue", "steelblue4"))
g2 <- g2 + labs(y = "Frecuencia") + ggtitle("Barras agrupadas")
g2 <- g2 + theme_pubr()

# # Crear gráfico de barras segmentadas estandarizado.
g3 <- ggplot(contingencia, aes(fill = Motor, y = Freq, x = Cambios))
g3 <- g3 + geom_bar(position = "fill", stat = "identity")
g3 <- g3 + scale_fill_manual(values=c("steelblue", "steelblue4"))
g3 <- g3 + labs(y = "Frecuencia") + ggtitle("Barras estandarizadas")
g3 <- g3 + theme_pubr()

# Crear una figura que contenga los tres gráficos.
g <- ggarrange(g1, g2, g3, nrow = 1, common.legend = TRUE)

# Agregar un título común en negrita y con fuente de 24 puntos.
titulo <- text_grob("Tipo de motor por cantidad de cambios",
                    size = 18)
g <- annotate_figure(g, top = titulo)

print(g)
```

Similar al gráfico de barras para dos variables, el **gráfico de mosaico** permite representar una tabla de contingencia. Para ello, divide un área en regiones y el área de cada región es proporcional al porcentaje de observaciones que representa. La figura 2.9, creada con el script 2.15 ejemplifica el uso de este tipo de gráficos, usando para ello las variables `Cambios` y `Motor`. En ella, el ancho de cada columna es proporcional a la cantidad de automóviles que tienen la correspondiente cantidad de cambios, mientras que la altura de cada barra de las columnas refleja la proporción de automóviles con un determinado tipo de motor.

Si nos fijamos bien en la figura 2.9, podemos ver claramente que los vehículos con 5 cambios son, por mucho, los menos frecuentes y que los con 3 cambios son algo más frecuentes que los que tienen 4 cambios. Del mismo modo, podemos ver que, para vehículos de 3 y 5 cambios, la proporción de vehículos con motor recto es la

misma, y mucho menor que la de aquellos con motor en forma de V. Sin embargo, este último no es muy frecuente en automóviles con 4 cambios.

Cabe destacar que, para este tipo de gráfico, se requiere emplear el paquete `ggmosaic`.

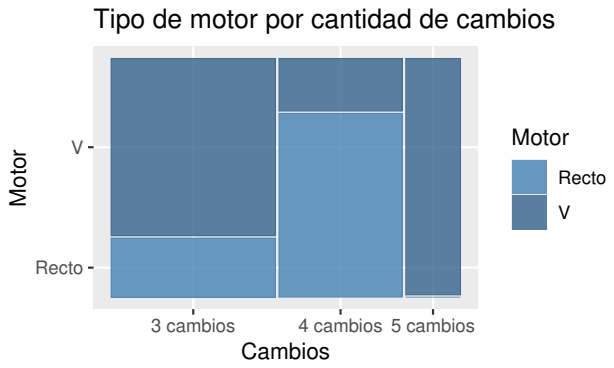


Figura 2.9: gráfico de mosaico para las variables `Cambios` y `Motor`.

Script 2.15: gráfico de mosaico para las variables `Cambios` y `Motor`.

```
library(ggmosaic)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

# Crear tabla de contingencia para las variables Cambios y Motor,
# y guardarla como data frame.
tabla <- xtabs(~ Cambios + Motor, data = datos)
contingencia <- as.data.frame(tabla)

# Crear gráfico de mosaico.
g <- ggplot(data = contingencia)
g <- g + geom_mosaic(aes(weight = Freq, x = product(Cambios), fill = Motor))
g <- g + labs(y = "Motor", x = "Cambios",
              title = "Tipo de motor por cantidad de cambios")
g <- g + scale_fill_manual(values=c("steelblue", "steelblue4"))

print(g)
```

2.2.5 Una variable numérica y otra categórica

Desde luego, también es importante poder comparar diferentes grupos de observaciones de acuerdo a una característica categórica, para lo cual los gráficos pueden ser de gran ayuda. Por ejemplo, la figura 2.10, creada mediante el script 2.16, muestra un gráfico de cajas para la variable `Rendimiento` agrupada por el número de cambios de los automóviles.

Script 2.16: gráfico de cajas por grupo.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

g <- ggboxplot(datos, x = "Cambios", y = "Rendimiento",
```

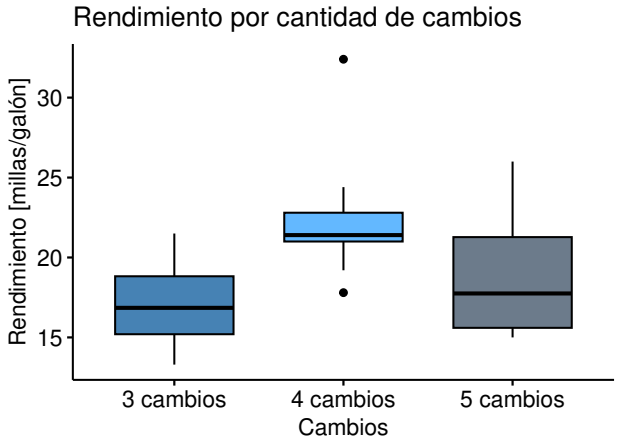


Figura 2.10: gráfico de cajas por grupo.

```
            palette = c("steelblue", "steelblue1", "slategray4"),
            fill = "Cambios",
            title = "Rendimiento por cantidad de cambios",
            xlab = "Cambios",
            ylab = "Rendimiento [millas/galón]")
g <- ggpar(g, legend = "none")

print(g)
```

Una buena alternativa, si la cantidad de observaciones es pequeña, es el **gráfico de tiras**, similar al gráfico de dispersión pero separando puntos de acuerdo a la variable categórica considerada. El script 2.17 construye este gráfico para la variable `Rendimiento` agrupada según los niveles de la variable `Cambios`, obteniéndose como resultado la figura 2.11.

Script 2.17: gráfico de tiras.

```
library(ggpubr)

# Cargar datos.
datos <- read.csv2("Mtcars.csv", stringsAsFactors = TRUE,
                   row.names = 1)

g <- ggstripchart(datos, x = "Cambios", y = "Rendimiento",
                  palette = c("steelblue", "steelblue1", "slategray4"),
                  color = "Cambios",
                  title = "Rendimiento por cantidad de cambios",
                  xlab = "Cambios",
                  ylab = "Rendimiento [millas/galón]")
g <- ggpar(g, legend = "none")

print(g)
```

2.3 EJERCICIOS PROPUESTOS

1. Averigua qué es un gráfico de puntos. ¿Cuándo es apropiado utilizar este tipo de gráfico para revisar datos? ¿En qué se diferencia de un gráfico de tiras?
2. Da un ejemplo de una variable en que la mediana la caracteriza mejor que la media.

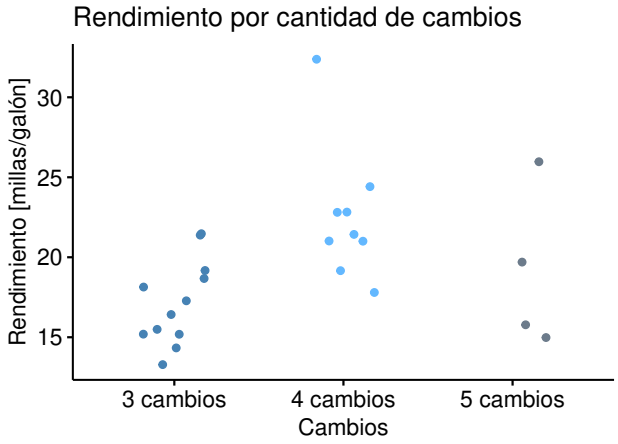
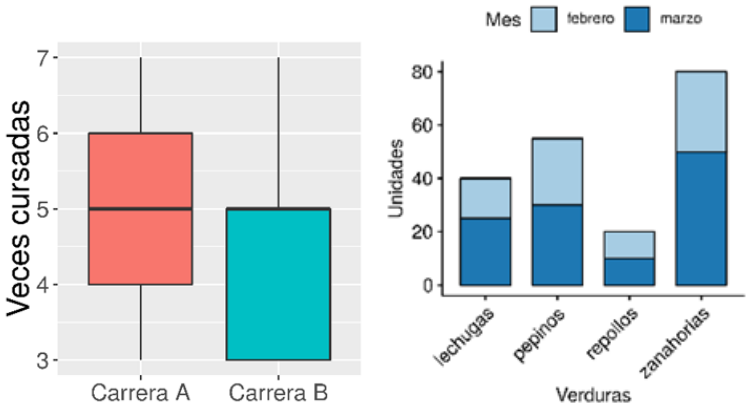


Figura 2.11: gráfico de tiras.

3. Da ejemplos de tres variables que posiblemente tengan una distribución simétrica, con asimetría positiva y con asimetría negativa, respectivamente, justificando bien cada caso.
4. Describe un estudio en que posiblemente los datos recolectados tengan una distribución bimodal.
5. ¿Cuál es la información más importante que nos podría dar un gráfico de dispersión?
6. ¿Por qué es importante conocer una medida de dispersión (variabilidad) de un conjunto de datos? Da un ejemplo novedoso para clarificar tu respuesta.
7. Considera la representación de la figura 2.12a de los datos obtenidos al tomar muestras aleatorias de estudiantes de dos carreras de la Facultad de Ingeniería para estudiar si el número de veces que se cursan las tres asignaturas de física contempladas en los planes de estudio depende o no de la carrera estudiada. Compara las distribuciones de ambos grupos. ¿En qué se parecen y en qué se diferencian?



(a) Ejercicio 7. (b) Ejercicio 8.

Figura 2.12: gráficos para los ejercicios propuestos.

8. El gráfico de la figura 2.12b muestra las unidades de verduras vendidas en uno de los kioscos cercanos a la Universidad durante el segundo y tercer mes del año pasado. Construye la tabla de contingencia correspondiente a los datos que se representan. ¿Qué mes tuvo mayores ventas? ¿En qué proporción?

Resuelve los siguientes ejercicios en R considerando para ello el conjunto de datos nativo `chickwts`.

9. ¿Cómo obtener los cuartiles para los pesos de los pollitos reportados en la columna `weight`? ¿Y cómo obtenerlos por cada tipo de alimento en la columna `feed`?

10. ¿Cómo se obtiene un gráfico de cajas para comparar los pesos de los pollitos por tipo de alimento suministrado?
11. ¿Cómo obtener un histograma de los pesos de los pollitos? ¿Cómo obtenerlos separados y con colores distintos por cada tipo de alimento?
12. ¿Cómo obtener un gráfico de densidad de los pesos de los pollitos? ¿Cómo obtenerlos separados en diferentes paneles por tipo de alimento?

2.4 BIBLIOGRAFÍA DEL CAPÍTULO

Diez, D., Barr, C. D., & Çetinkaya-Rundel, M. (2017). *OpenIntro Statistics* (3.ª ed.). https://www.openintro.org/book/os/.

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. SAGE Publications Ltd.

STDHA. (s.f.). Descriptive Statistics and Graphics - Easy Guides - Wiki - STHDA. Consultado el 31 de marzo de 2021, desde http://www.sthda.com/english/wiki/descriptive-statistics-and-graphics