INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.

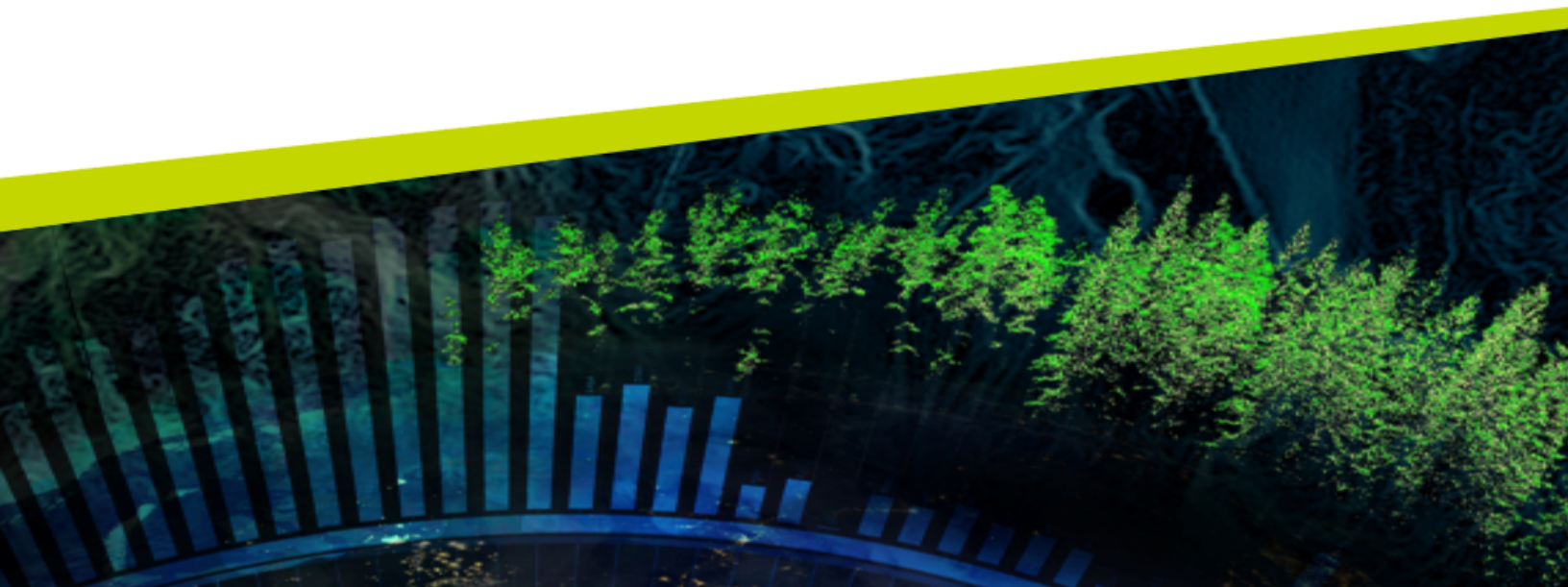

Tabla de contenido

Índice de tablas

Índice de figuras

Índice de scripts

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. Te sugerimos, entonces, que lo instales junto con el entorno de desarrollo integrado RStudio.

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 (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	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
⋮	⋮	⋮	⋮	⋮	⋮	⋮
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 a su rango de valores o a un listado de valores posibles.

Variable	Descripción
id	Identificador de la observación.
género	Género del estudiante (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, C: Carabineros).

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

Si estudias 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` de 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` de la tabla 1.1.

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, aunque para ello es necesario que la variable tenga 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 completa", "media", "superior" y "postgrado". Aquí podemos apreciar claramente que los niveles están ordenados de manera creciente.
- **Escala de intervalo**: 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. El cero está dado por el punto de congelación del agua. La medida o tamaño se calcula en base a los puntos de congelación y ebullición del agua. Sin embargo, 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, o 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. Sin embargo, existe una transformación lineal que nos permite transformar una medida en una escala a su equivalente en otra escala.
- **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, por ejemplo, las que permiten medir la masa o la distancia. En una escala de razón, la diferencia entre dos puntos es independiente de la unidad de medida. Por ejemplo, si medimos la masa de dos objetos, la razón es constante independientemente de si empleamos kilogramos, libras u onzas (a diferencia de lo que ocurre con la temperatura usando las escalas Celsius y Fahrenheit).

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.

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 RStudio (2021). 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 o 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.

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

```
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.

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.
20 head(datos1)
21
22 # Mostrar las últimas 6 filas del conjunto de datos
23 # almacenado en datos1.
24 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.

```
1  # Instalar un paquete.
2  install.packages("ggpubr")
3
4  # Primera forma de importar un paquete.
5  library(ggpubr)
6
7  # Segunda forma de importar un paquete.
8  require(ggplot2)
9
10 # Importar un paquete, instalándolo de ser necesario.
11 if(!require(dplyr)){
12   install.packages("dplyr", dependencies=TRUE)
13   require(dplyr)
14 }
```

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 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 un dataframe.

```
# 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, "C:/Inferencia/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("C:/Inferencia/Ejemplo.csv", stringsAsFactors = FALSE)

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

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

```

```r
# 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.
dataframe <- rbind(dataframe, nueva)

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

# Guardar el dataframe en un archivo csv .
write.csv2(dataframe, "C:/Inferencia/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 a su valor.
- `arrange()`: modifica el orden de las filas.
- `select()`: permite seleccionar variables (características) por sus nombres, a la vez que las reordena.
- `mutate()`: permite 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`.

```r
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* y definido en el paquete `magrittr`, 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).

Otra transformación que se usa a menudo es la de pivotar la matriz de datos, cuyo efecto es el de "alrargar" 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 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))
cat("\n")

```

```r
# 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.
- `labels`: argumento opcional con las etiquetas asociadas a cada valor.
- `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.

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`.

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

```
library(dplyr)

# Cargar conjunto de datos.
datos <- mtcars

# 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
# 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 , "C:/Inferencia/Mtcars.csv")
```

1.3.5 Fórmulas

Si bien hasta ahora solo tenemos una definición preliminar de lo que es un modelo estadístico, necesitamos conocer una herramienta para representarlos en R, pues son una parte fundamental del funcionamiento de este lenguaje.

Para entender de manera sencilla qué es una fórmula, podemos simplemente decir que permite capturar una expresión no evaluada, y que está asociada a un ambiente. Su sintaxis básica tiene la forma `variable independiente` $\sim$ `variables dependientes`, lo que nos indica, entonces, que las fórmulas representan una relación entre variables.

Tomemos una vez más el conjunto de datos `iris`. Podríamos representar la asociación entre la especie de iris (variable independiente) y las dimensiones de sus pétalos (variables dependientes) como `Species` $\sim$ `Petal.Length + Petal.Width`.

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

1.4 EJERCICIOS PROPUESTOS

1. 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.
   a) ¿Cómo sería una matriz de datos para este estudio? Muestra algunas filas de ella como ejemplos.
   b) ¿Cuál podría ser la población objetivo?
   c) ¿Qué se entendería por unidad de observación?
   d) ¿Qué tipo de variable sería "el número de horas dedicadas a distraerse después de la jornada laboral usual" que respondió cada persona entrevistada?
   e) ¿Existe alguna variable categórica? Si es así, ¿de qué tipo? ¿Con qué niveles?
   f) ¿Qué dato(s) correspondería(n) a un estadístico?
   g) ¿Cuál(es) sería(n) el(los) parámetro(s) en estudio?
   h) ¿Logra el estudio establecer que ser mujer chilena ocasiona tener menos horas dedicadas a distraerse después de la jornada laboral usual?
2. 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()`.
3. 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.
4. 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.
5. 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 la variable `pollos`?
   b) ¿Cómo se ve la estructura de la matriz de datos almacenada en `pollos`?
6. 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`.

REFERENCIAS

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/

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.

RStudio. (2021). RStudio. Consultado el 2 de abril de 2021, desde https://rstudio.com/products/rstudio/

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/