

Capítulo 2

Carga y manipulación de datos

Ahora pasamos a usar R para realizar análisis de datos. Nuestros primeros pasos básicos son simplemente cargar datos en R y limpiar los datos para que se adapten a nuestros propósitos. La limpieza y recodificación de datos son a menudo una tarea tediosa de análisis de datos, pero sin embargo son esenciales porque los datos mal codificados producirán resultados erróneos cuando se estima un modelo usándolos. (En otras palabras, basura entra, basura sale). En este capítulo, cargaremos varios tipos de datos usando diferentes comandos, veremos nuestros datos para comprender sus características, practicaremos recodificar datos para limpiarlos según sea necesario, fusionar datos conjuntos y remodelar conjuntos de datos.

Nuestro ejemplo de trabajo en este capítulo será un subconjunto de Poe et al. (1999) Los datos de Political Terror Scale sobre derechos humanos, que son una actualización de los datos de Poe y Tate (1994). Mientras que sus datos completos cubren 1976-1993, nos centramos únicamente en el año 1993. Las ocho variables que contiene este conjunto de datos son:

país: Una variable de carácter que enumera el país por nombre.

democ: Puntuación del país en la escala de democracia Polity III. Las puntuaciones van desde 0 (menos democrático) a 10 (más democrático).

sdnew: La escala de terror político del Departamento de Estado de EE. UU. Las puntuaciones van desde 1 (terrorismo de estado bajo, menor número de violaciones de la integridad personal) al 5 (mayor número de violaciones de la integridad personal).

militar: Una variable ficticia codificada 1 para un régimen militar, 0 en caso contrario.

gnpcats: Nivel de PNB per cápita en cinco categorías: 1 = menos de \$ 1000, 2 = \$ 1000–\$ 1999, 3 = \$ 2000–\$ 2999, 4 = \$ 3000–\$ 3999, 5 = más de \$ 4000.

lpop: Logaritmo de la población nacional.

civ_war: Una variable ficticia codificada con 1 si está involucrada en una guerra civil, 0 en caso contrario.

int_war: Una variable ficticia codificada como 1 si está involucrada en una guerra internacional, 0 en caso contrario.

Electrónico suplementario material: La en línea versión de esto capítulo (doi: [10.1007/978-3-319-23446-5_2](https://doi.org/10.1007/978-3-319-23446-5_2)) contiene usuarios autorizados material, que está disponible para suplementarios.

2.1 Lectura de datos

Ingresa datos R es bastante fácil. Hay tres formas principales de importar datos: ingresar los datos manualmente (quizás escritos en un archivo de script), leer datos de un archivo con formato de texto e importar datos de otro programa. Dado que es un poco menos común en Ciencias Políticas, el ingreso de datos manualmente se ilustra en los ejemplos del Cap.10, en el contexto de la creación de vectores, matrices y marcos de datos. En esta sección, nos enfocamos en importar datos de archivos guardados.

Primero, consideramos cómo leer en un archivo de texto delimitado con la `read.table` comando. R leerá en una variedad de archivos delimitados. (¿Para todas las opciones asociadas con este tipo de comando? `read.table` en R.) En los datos basados en texto, normalmente cada línea representa una observación única y algún delimitador designado separa cada variable de la línea. El predeterminado para `read.table` es un archivo delimitado por espacios en el que cualquier espacio en blanco designa variables diferentes. Dado que nuestro Poe et al. archivo de datos, llamado `hmnrghts.txt`, está separado por espacios, podemos leer nuestro archivo en R usando la siguiente línea de código. Este archivo de datos está disponible en el Dataverse mencionado en la página vii o en el enlace de contenido del capítulo en la página 13. Es posible que deba utilizar `setwd` comando introducido en el Cap. 1 apuntar R a la carpeta donde ha guardado los datos. Después de esto, ejecute el siguiente código:

```
hmnrghts <- read.table("hmnrghts.txt",
  encabezado = VERDADERO, na = "NA")
```

Nota: Como se mencionó en el capítulo anterior, R permite al usuario dividir un solo comando en varias líneas, lo que hemos hecho aquí. A lo largo de este libro, los comandos que abarcan varias líneas se distinguirán con sangría francesa. Pasando al código en sí, observe algunas características: Una, como se señaló en el capítulo anterior, el símbolo de la flecha izquierda (`<-`) asigna nuestro archivo de entrada a un objeto. Por eso, `hmnrghts` es el nombre que asignamos a nuestro archivo de datos, pero podríamos haberlo llamado de cualquier forma. En segundo lugar, el primer argumento de `read.table` comando llama al nombre del archivo de texto `hmnrghts.txt`. Podríamos haber precedido a este argumento con la `archivo =` opción, y habríamos tenido que hacerlo si no hubiéramos incluido esto como el primer argumento, pero R reconoce que el archivo en sí es normalmente el primer argumento que toma este comando. En tercer lugar, especificamos `encabezado = VERDADERO`, lo que transmite que la primera fila de nuestro archivo de texto enumera los nombres de nuestras variables. Es esencial que este argumento se identifique correctamente, de lo contrario, los nombres de variables pueden asignarse erróneamente como datos o los datos como nombres de variables.¹ Finalmente, dentro del archivo de texto, los caracteres N / A se escriben siempre que falta una observación de una variable. La opción `na = "NA"` transmite a R que este es el símbolo del conjunto de datos para un valor faltante. (Otros símbolos comunes de ausencia son un punto (.) O el número -9999.)

El comando `read.table` también tiene otras opciones importantes. Si su archivo de texto usa un delimitador que no sea un espacio, entonces esto se puede transmitir a R utilizando la `sep` opción. Por ejemplo, incluyendo `sep = "\t"` en el comando anterior habría

¹Una mirada más cercana al archivo en sí mostrará que nuestra línea de encabezado de nombres de variables en realidad tiene un elemento menos que cada línea de datos. Cuando este es el caso, R supone que el primer elemento de cada línea es un índice de observación. Dado que eso es cierto en este caso, nuestros datos se leen correctamente.

nos permitió leer en un archivo de texto separado por tabulaciones, mientras `sep = ","` habría permitido un archivo separado por comas. Los comandos `read.csv` y `read.delim` son versiones alternativas de `read.table` que simplemente tienen diferentes valores predeterminados. (Particularmente, `read.csv` está orientado a **leer comma-separado** **archivos** de **alues** y `read.delim` está orientado a **leer pestaña-delimitar** **archivos** incluidos, aunque algunos otros valores predeterminados también cambian.) Otra opción importante para estos comandos es `quote`. Los valores predeterminados varían en estos comandos para los cuales los caracteres designan variables basadas en cadenas que usan texto alfabético como valores, como el nombre de la observación (por ejemplo, país, estado, candidato). El comando, por defecto, usa comillas simples o dobles alrededor de la entrada en el archivo de texto. Esto sería un problema si se usaran comillas dobles para designar el texto, pero hubiera apóstrofes en el texto. Para compensar, simplemente especifique la opción `quote = "\""` para permitir solo comillas dobles. (Observe la barra invertida para indicar que la comilla doble es un argumento). Alternativamente, `read.csv` y `read.delim` Ambos solo permiten comillas dobles de forma predeterminada, por lo que se especifica `quote = "\""` permitiría comillas simples o dobles, o `quote = "'"` cambiaría a comillas simples. Los autores también pueden especificar otros caracteres en esta opción, si es necesario.

Una vez que descargue un archivo, tiene la opción de especificar el directorio de ruta completo en el comando para abrir el archivo. Supongamos que hubiéramos salvado `hmnrghts.txt` en el directorio de ruta `C:/temp/`, entonces podríamos cargar el archivo de la siguiente manera:

```
hmnrghts <- read.table("C:/temp/hmnrghts.txt",
  encabezado = VERDADERO, na = "NA")
```

Como se mencionó cuando cargamos el archivo por primera vez, otra opción habría sido usar la `setwd` comando para **colocar la working Directory**, lo que significa que no tendríamos que enumerar la ruta completa del archivo en la llamada a `read.table`. (Si hacemos esto, todos los archivos de entrada y salida irán a este directorio, a menos que especifiquemos lo contrario). Finalmente, en cualquier sistema basado en GUI (por ejemplo, no terminal), podríamos escribir:

```
hmnrghts <- read.table(file.choose(), header = TRUE, na = "NA")
```

La `file.choose()` La opción abrirá un navegador de archivos que permitirá al usuario localizar y seleccionar el archivo de datos deseado, que R luego se asignará al objeto nombrado en la memoria (`hmnrghts` en este caso). Esta opción de navegador es útil en el análisis interactivo, pero menos útil para programas automatizados.

Otro formato de datos basados en texto es un *archivo de ancho fijo*. Los archivos de este formato no utilizan un carácter para delimitar variables dentro de una observación. En cambio, ciertas columnas de texto están constantemente dedicadas a una variable. Por lo tanto, R necesita saber qué columnas definen cada variable para leer en los datos. El comando para **leer** un **Fijo width File** es `read.fwf`. Como una ilustración rápida de cómo cargar este tipo de datos, cargaremos un conjunto de datos diferente: votaciones nominales del 113 ° Senado de los Estados Unidos, el período que se extiende desde 2013 hasta 2015.² Este conjunto de datos se revisará en un problema de práctica.

²Estos datos fueron recopilados por Jeff Lewis y Keith Poole. Para más información, ver <http://www.voteview.com/senate113.htm>.

en el Cap. 8. Para abrir estos datos, comience por descargar el archivo `sen113kh.ord` del Dataverse que aparece en la página vii o del enlace del contenido del capítulo en la página 13. Luego, escriba:

```
senate.113 <-read.fwf("sen113kh.ord",
  anchos = c(3,5,2,2,8,3,1,1,11, rep(1,657)))
```

El primer argumento de `read.fwf` es el nombre del archivo, que extraemos de una URL. (La extensión del archivo es `.ord`, pero el formato es texto sin formato. Intente abrir el archivo en el Bloc de notas o TextEdit solo para familiarizarse con el formato.) El segundo argumento, `anchos`, es esencial. Para cada variable, debemos ingresar el número de caracteres asignados a esa variable. En otras palabras, la primera variable tiene tres caracteres, la segunda tiene cinco caracteres y así sucesivamente. Este procedimiento debería dejar en claro que debemos tener un libro de códigos para un archivo de ancho fijo, o ingresar los datos es un esfuerzo inútil. Observe que el último componente de `anchos` el argumento es `rep(1,657)`. Esto significa que nuestro conjunto de datos termina con 657 variables de un carácter. Estos son los 657 votos que emitió el Senado durante ese período del Congreso, con cada variable registrando si cada senador votó sí, no, presente o no votó.

Con cualquier tipo de archivo de datos, incluido el ancho fijo, si el archivo en sí no tiene los nombres de las variables, podemos agregarlos en R. (Nuevamente, aquí es útil un buen libro de códigos). `read.table`, `read.csv`, y `read.fwf` todos incluyen una opción llamada `nombres de col.` que permite al usuario nombrar cada variable en el conjunto de datos al leer en el archivo. Sin embargo, en el caso de las votaciones nominales del Senado, es más fácil para nosotros nombrar las variables después de la siguiente manera:

```
colnames(senate.113)[1:9] <- c("congreso", "icpsr", "código de estado",
  "cd", "state.name", "party", "occupancy", "attaining", "name") for(i in 1: 657)
{colnames(senate.113)[i + 9] <- pegar ("RC", i, sep = "")}
```

En este caso, usamos el `colnames` comando para establecer los nombres de las variables. A la izquierda de la flecha, especificando `[1: 9]`, indicamos que solo estamos nombrando las primeras nueve variables. A la derecha de la flecha, usamos uno de los *mas fundamental* comandos en R: la **C**ombine comando (`C`), que combina varios elementos en un vector. En este caso, nuestro vector incluye los nombres de las variables en texto. En la segunda línea, procedemos a nombrar los 657 votos nominales. `RC1`, `RC2`, . . . , `RC657`.

Para ahorrar tipeo, hacemos esta asignación usando un `for` loop, que se describe con más detalle en el Cap. 11. Dentro de `for` bucle, usamos el `pegar` comando, que simplemente imprime nuestro texto ("`RC` ") y el número de índice `i`, `sep` clasificado por nada (de ahí las comillas vacías al final). Por supuesto, por defecto, R asigna nombres de variables genéricas (`V1`, `V2`, etc.), por lo que un lector que se contente con usar nombres genéricos puede omitir este paso, si lo prefiere. (Tenga en cuenta, sin embargo, que si nombramos las primeras nueve variables como lo hicimos, la primera votación nominal se llamaría `V10` sin que apliquemos un nuevo nombre).

2.1.1 Lectura de datos de otros programas

Volviendo a nuestro ejemplo de derechos humanos, también puede importar datos de muchos otros programas estadísticos. Una de las bibliotecas más importantes de R es el extranjero, lo que facilita la incorporación de datos de otros paquetes estadísticos, como SPSS, Stata y Minitab.³ Como alternativa a la versión de texto de los datos de derechos humanos, también podríamos cargar un archivo de datos con formato Stata, `hmnrghts.dta`.

Los archivos de estadísticas generalmente tienen una extensión de archivo de `dta`, que es lo que el `read.dta` comando se refiere. (Similar, `read.spss` voluntad leer un SPSS-archivo formateado con el `.sav` extensión de archivo.) Para abrir nuestros datos en formato Stata, necesitamos descargar el archivo `hmnrghts.dta` desde el Dataverse vinculado en la página vii o el contenido del capítulo vinculado en la página 13. Una vez que lo guardamos en nuestro disco duro, podemos configurar nuestro directorio de trabajo, listar la ruta completa del archivo o usar el `file.choose()` comando para acceder a nuestros datos. Por ejemplo, si descargamos el archivo, que se llama `hmnrghts.dta`, en nuestro C: \ temp \ carpeta, podríamos abrirla escribiendo:

```
biblioteca(extranjera)
setwd("C: / temp /")
hmnrghts.2 <- read.dta("hmnrghts.dta")
```

Cualquier dato en formato Stata que seleccione se convertirá a R formato. Una palabra de advertencia, por defecto si hay etiquetas de valor en datos con formato Stata, R importará las etiquetas como una variable con formato de cadena. Si esto es un problema para usted, intente importar los datos sin etiquetas de valor para guardar las variables usando los códigos numéricos. Vea el comienzo del Cap.7 para un ejemplo de la `convert.factors = FALSE` opción. (Una opción para los conjuntos de datos que no son excesivamente grandes es cargar dos copias de un conjunto de datos de Stata, una con las etiquetas como texto para que sirva como un libro de códigos y otra con códigos numéricos para el análisis). Siempre es bueno ver exactamente cómo los datos se formatean inspeccionando la hoja de cálculo después de importar con las herramientas descritas en la Sección.2.2.

2.1.2 Tramas de datos en R

R distingue entre *vectores*, *listas*, *marcos de datos*, y *matrices*. Cada uno de estos es un objeto de una clase diferente en R. Los vectores están indexados por longitud y las matrices están indexadas por filas y columnas. Las listas no son omnipresentes para los análisis básicos, pero son útiles para el almacenamiento complejo y, a menudo, se las considera como *genérico* vectores donde cada elemento puede ser cualquier clase de objeto. (Por ejemplo, una lista podría ser un vector de

³La extranjero El paquete se usa con tanta frecuencia que ahora se descarga con cualquier R instalación. Sin embargo, en el improbable caso de que el paquete no se cargue con el `Biblioteca` comando, simplemente escriba `install.packages("extranjer")` en el símbolo del sistema para descargarlo. Alternativamente, para los usuarios que deseen importar datos de *Sobresalir*, hay dos opciones: una es guardar el archivo de Excel en formato de valores separados por comas y luego usar el `read.csv` mando. El otro es instalar el `XLConnect` biblioteca y use la `readWorksheetFromFile` mando.

resultados del modelo, o una combinación de marcos de datos y mapas.) Un marco de datos es una matriz que R designa como un conjunto de datos. Con un marco de datos, las columnas de la matriz pueden denominarse variables. Después de leer un conjunto de datos, R tratará sus datos como un marco de datos, lo que significa que puede hacer referencia a cualquier variable dentro de un marco de datos agregando `$ VARIABLENAME` al nombre del marco de datos.⁴ Por ejemplo, en nuestros datos de derechos humanos podemos imprimir la variable país para ver qué países están en el conjunto de datos:

```
hmnrghts $ país
```

Otra opción para llamar a variables, aunque un *desaconsejable* uno, es usar el adjuntar mando. R permite al usuario cargar múltiples conjuntos de datos a la vez (en contraste con algunos de los programas de análisis de datos disponibles comercialmente). Laaduntar El comando coloca un conjunto de datos al frente y permite al usuario llamar directamente los nombres de las variables sin hacer referencia al nombre del marco de datos. Por ejemplo:

```
adjuntar (hmnrghts)
país
```

Con este código, R reconocería país de forma aislada como parte del conjunto de datos adjunto e imprímalo como en el ejemplo anterior. El problema con este enfoque es queR puede almacenar objetos en la memoria con el *mismo nombre* como algunas de las variables del conjunto de datos. De hecho, al recodificar datos, el usuario debesiempre referirse al marco de datos por su nombre, de lo contrario R confusamente creará una copia de la variable en la memoria que es distinta de la copia en el marco de datos. Por esta razón, generalmente recomiendo no adjuntar datos. Si, por alguna circunstancia, un usuario siente que adjuntar un marco de datos es inevitable, entonces el usuario puede realizar lo que debe hacerse con los datos adjuntos y luego usar eldespegar comando lo antes posible. Este comando funciona como era de esperar, eliminando la designación de un marco de datos de trabajo y ya no permite que el usuario llame a las variables de forma aislada:

```
separar (hmnrghts)
```

2.1.3 Escritura de datos

Para exportar los datos que está utilizando en R a un archivo de texto, use las funciones escribir.tabla o write.csv. Dentro de extranjero Biblioteca, write.dta permite al usuario escribir un archivo con formato Stata. Como ejemplo simple, podemos generar una matriz con cuatro observaciones y seis variables, contando de 1 a 24. Luego, podemos escribir esto en un archivo de valores separados por comas, un archivo de texto delimitado por espacios y un archivo Stata:

```
x <- matriz (1:24, nrow = 4) write.csv (x, file = "sample.csv")
write.table (x, file = "sample.txt") write.dta (as.data.frame (x),
archivo = "sample.dta")
```

⁴Más técnicamente, los marcos de datos son objetos del S3 clase. Para todosS3 objetos, los atributos del objeto (como las variables) se pueden llamar con el signo de dólar (\$).

Tenga en cuenta que el comando `as.data.frame` convierte matrices en marcos de datos, una distinción descrita en la sección anterior. El comando `write.dta` espera que el objeto sea del marco de datos clase. No es necesario realizar esta conversión si el objeto ya está formateado como datos, en lugar de una matriz. Para comprobar su esfuerzo para guardar los archivos, intente eliminar `X` de la memoria con el comando `rm (x)` y luego restaurar los datos de uno de los archivos guardados.

Para mantenerse al día con el destino de los archivos de datos guardados, los archivos que escribimos se guardarán en nuestro *directorio de trabajo*. A **obtener** la **working Directory** (es decir, tener R dinos qué es), simplemente escribe: `getwd ()`. Para cambiar el directorio de trabajo donde se escribirán los archivos de salida, podemos **colocar la working Directory**, usando el mismo `setwd` comando que consideramos al abrir archivos anteriormente. Todos los archivos guardados posteriormente se enviarán al directorio especificado, a menos que R se dice explícitamente lo contrario.

2.2 Visualización de atributos de los datos

Una vez que se ingresan los datos en R, la primera tarea debería ser inspeccionar los datos y asegurarse de que se hayan cargado correctamente. Con un conjunto de datos relativamente pequeño, simplemente podemos imprimir todo el marco de datos en la pantalla:

`hmnrghts`

La impresión del conjunto de datos completo, por supuesto, no se recomienda ni es útil con conjuntos de datos grandes. Otra opción es mirar los nombres de las variables y las primeras líneas de datos para ver si los datos están estructurados correctamente a través de algunas observaciones. Esto se hace con el comando:

`cabeza (hmnrghts)`

Para obtener una lista rápida de los nombres de las variables en nuestro conjunto de datos (que también puede ser útil si se olvida la ortografía exacta o el uso de mayúsculas en los nombres de las variables) escriba:

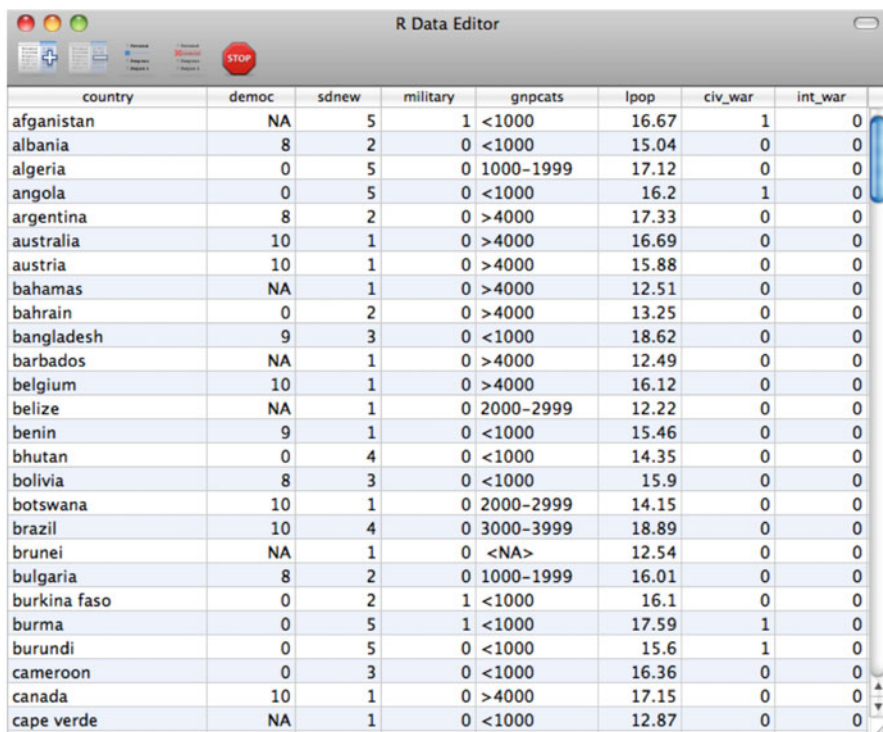
`nombres (hmnrghts)`

Una ruta para obtener una visión completa de los datos es utilizar el comando:

`arreglar (hmnrghts)`

Esto presenta los datos en una hoja de cálculo que permite una vista rápida de las observaciones o variables de interés, así como la oportunidad de ver que la matriz de datos se cargó correctamente. Un ejemplo de esta ventana del editor de datos que `arreglar` abre se presenta en la Fig. 2.1. El usuario tiene la opción de editar datos dentro de la ventana de la hoja de cálculo que `arreglar` crea, aunque a menos que los datos revisados se escriban en un nuevo archivo, no habrá un registro permanente de estos cambios.⁵ Además, es clave tener en cuenta que antes de continuar una sesión

⁵La Vista el comando es similar a `reparar`, pero no permite la edición de observaciones. Si prefiere solo poder ver los datos sin editar valores (tal vez incluso apoyándose accidentalmente en su teclado), entonces `Vista` podría ser preferible.



country	democ	sdnew	military	gnpcats	lpop	civ_war	int_war
afghanistan	NA	5	1	<1000	16.67	1	0
albania	8	2	0	<1000	15.04	0	0
algeria	0	5	0	1000-1999	17.12	0	0
angola	0	5	0	<1000	16.2	1	0
argentina	8	2	0	>4000	17.33	0	0
australia	10	1	0	>4000	16.69	0	0
austria	10	1	0	>4000	15.88	0	0
bahamas	NA	1	0	>4000	12.51	0	0
bahrain	0	2	0	>4000	13.25	0	0
bangladesh	9	3	0	<1000	18.62	0	0
barbados	NA	1	0	>4000	12.49	0	0
belgium	10	1	0	>4000	16.12	0	0
belize	NA	1	0	2000-2999	12.22	0	0
benin	9	1	0	<1000	15.46	0	0
bhutan	0	4	0	<1000	14.35	0	0
bolivia	8	3	0	<1000	15.9	0	0
botswana	10	1	0	2000-2999	14.15	0	0
brazil	10	4	0	3000-3999	18.89	0	0
brunei	NA	1	0	<NA>	12.54	0	0
bulgaria	8	2	0	1000-1999	16.01	0	0
burkina faso	0	2	1	<1000	16.1	0	0
burma	0	5	1	<1000	17.59	1	0
burundi	0	5	0	<1000	15.6	1	0
cameroon	0	3	0	<1000	16.36	0	0
canada	10	1	0	>4000	17.15	0	0
cape verde	NA	1	0	<1000	12.87	0	0

Figura 2.1 R editor de datos abierto con el reparar mando

con más comandos, debe cerrar la ventana del editor de datos. La consola está congelada mientras elreparar la ventana está abierta.

Hablaremos más sobre estadística descriptiva en el Cap. 4. Mientras tanto, sin embargo, Puede ser informativo ver algunos de los estadísticos descriptivos básicos (incluida la media, la mediana, el mínimo y el máximo), así como un recuento del número de observaciones faltantes para cada variable:

```
resumen(hmnrghts)
```

Alternativamente, esta información se puede obtener solo para una sola variable, como la población registrada:

```
resumen(hmnrghts $ lpop)
```

2.3 Declaraciones lógicas y generación de variables

A medida que pasamos a limpiar los datos que se cargan en R, un conjunto de herramientas esencial es el grupo de declaraciones lógicas. Declaraciones lógicas (o booleanas) enR son evaluados en cuanto a si son CIERTO o FALSO. Mesa 2.1 resume los operadores lógicos comunes en R.

Cuadro 2.1 Operadores lógicos en R

Operador	Medio
<	Menos que
<=	Menor o igual a
>	Mayor que
> =	Mayor o igual a Igual a
==	
!=	No igual a
Y	Y
	O

Tenga en cuenta que la declaración booleana "es igual a" se designa con dos signos iguales (==), mientras que un solo signo igual (=) sirve como operador de asignación.

Para aplicar algunos de estos operadores booleanos de la tabla 2.1 en la práctica, supongamos, por ejemplo, que quisiéramos saber qué países estaban en una guerra civil y tenían un puntaje de democracia superior al promedio en 1993. Podríamos generar una nueva variable en nuestro conjunto de datos de trabajo, que llamaré dem.civaunque el usuario puede elegir el nombre). Luego, podemos ver una tabla de nuestra nueva variable y listar todos los países que se ajustan a estos criterios:

```
hmnrghts $ dem.civ <- as.numeric (hmnrghts $ civ_war == 1 &
                                hmnrghts $ democ> 5.3)
tabla (hmnrghts $ dem.civ) hmnrghts $ país
[hmnrghts $ dem.civ == 1]
```

En la primera línea, hmnrghts \$ dem.civ define nuestra nueva variable dentro del conjunto de datos de derechos humanos.⁶ A la derecha, tenemos una declaración booleana de dos partes: la primera pregunta si el país está en una guerra civil y la segunda pregunta si el puntaje de democracia del país es más alto que el promedio de 5.3. El ampersand (&) requiere que ambas declaraciones sean verdaderas simultáneamente para que toda la declaración sea verdadera. Todo esto está incrustado en el as.numeric comando, que codifica nuestra salida booleana **como a numérico** variable.

Específicamente, todos los valores deCIERTO se establecen en 1 y FALSO los valores se establecen en 0. Esta codificación suele ser más conveniente para fines de modelado. La siguiente línea nos da una tabla de las frecuencias relativas de 0 y 1. Resulta que solo cuatro países tenían niveles de democracia por encima del promedio y estuvieron involucrados en una guerra civil en 1993. Para ver qué países, la última línea preguntaR para imprimir los nombres de los países, pero los corchetes que siguen al vector indican qué observaciones imprimir: Solo las que puntúan 1 en esta nueva variable.⁷

⁶Sin embargo, tenga en cuenta que las nuevas variables que creamos, las observaciones que eliminamos o las variables que recodificamos solo cambian los datos en *memoria de trabajo*. Por lo tanto, nuestro archivo de datos original en disco permanece sin cambios y, por lo tanto, es seguro para la recuperación. Nuevamente, debemos usar uno de los comandos de Sect.2.1.3 si queremos guardar una segunda copia de los datos, incluidos todos nuestros cambios.

⁷La salida imprime los cuatro nombres de países y cuatro valores de N / A. Esto significa que en cuatro casos, una de las declaraciones de dos componentes fue CIERTO pero la otra declaración no se pudo evaluar porque faltaba la variable.

Otro tipo de declaración lógica en R que puede ser útil es el `is` declaración. Estas declaraciones preguntan si una observación u objeto cumple algún criterio. Por ejemplo, `is.na` es un caso especial que pregunta si falta una observación o no. Alternativamente, declaraciones como `is.matrix` o `is.data.frame` preguntan si un objeto es de cierta clase. Considere tres ejemplos:

```
tabla(is.na(hmnrghs $ democ))
is.matrix(hmnrghs)
is.data.frame(hmnrghs)
```

La primera declaración pregunta por cada observación si falta el valor de la democracia. La mesa El comando luego agrega esto y nos informa que faltan 31 observaciones. Las siguientes dos declaraciones preguntan si nuestro conjunto de datos `hmnrghs` se guarda como una matriz, luego como un marco de datos. La `is.matrix` declaración devuelve FALSO, lo que indica que los comandos basados en matrices no funcionarán en nuestros datos, y el `is.data.frame` declaración devuelve CIERTO, lo que indica que se almacena como un marco de datos. Con un sentido de declaraciones lógicas en R, ahora podemos aplicarlos a la tarea de limpiar los datos.

2.4 Limpieza de datos

Una de las primeras tareas de la limpieza de datos es decidir cómo lidiar con *datos perdidos*. R designa los valores perdidos con N / A. Traduce los valores perdidos de otros paquetes de estadísticas al N / A formato faltante. Independientemente de cómo un investigador maneje los datos faltantes, es importante tener en cuenta la proporción relativa de valores no observados en los datos y qué información se puede perder. Una opción (algo burda) para lidiar con la falta sería podar el conjunto de datos mediante la eliminación por lista, o eliminar todas las observaciones para las que no se registra una sola variable. Para crear un nuevo conjunto de datos que puede de esta manera, escriba:

```
hmnrghs.trim <- na.omit(hmnrghs)
```

Esto disminuye el número de observaciones de 158 a 127, por lo que se ha perdido una cantidad tangible de información.

La mayoría de los comandos de modelado en R Brinde a los usuarios la opción de estimar el modelo solo sobre observaciones completas, implementando la eliminación por lista sobre la marcha. Como advertencia, la eliminación por lista es en realidad el valor predeterminado en los comandos básicos para los modelos lineales y lineales generalizados, por lo que la pérdida de datos puede pasar desapercibida si el usuario no tiene cuidado. Se insta a los usuarios con una sólida formación en modelos basados en regresiones a que consideren métodos alternativos para tratar los datos faltantes que sean superiores a la eliminación por listas. En particular, el *ratones* y *Amelia* las bibliotecas implementan la útil técnica de la imputación múltiple (para obtener más información, consulte King et al. 2001; Little y Rubin 1987; Frotar 1987).

Si, por alguna razón, el usuario necesita volver a designar los valores faltantes como si tuvieran algún valor numérico, el `is.na` El comando puede ser útil. Por ejemplo, si fuera beneficioso enumerar los valores faltantes como 9999, entonces estos podrían codificarse como:

```
hmnrghs $ democ[is.na(hmnrghs $ democ)] <- -9999
```

En otras palabras, todos los valores de la democracia para los que falta el valor tomarán el valor de 9999. *Ten cuidado*, aunque, como R y todos sus comandos de modelado ahora considerarán el valor anteriormente perdido como una observación válida e insertarán el valor engañoso de 9999 en cualquier análisis. Este tipo de acción solo debe tomarse si se requiere para la gestión de datos, un tipo especial de modelo en el que los valores extraños se pueden tachar, o el raro caso en el que una observación faltante en realidad puede adquirir un valor significativo (por ejemplo, un presupuesto conjunto de datos donde los elementos faltantes representan un gasto de \$ 0).

2.4.1 Subconjuntos de datos

En muchos casos, es conveniente crear subconjuntos de nuestros datos. Esto puede significar que solo queremos observaciones de un cierto tipo, o puede significar que deseamos reducir el número de variables en el marco de datos, quizás porque los datos incluyen muchas variables que no son de interés. Si, en nuestros datos de derechos humanos, solo quisiéramos centrarnos en los países que tenían una puntuación de democracia de 6 a 10, podríamos llamar a este subconjunto `dem.rights` y créelo de la siguiente manera:

```
dem.rights <- subconjunto (hmnrghts, subconjunto = democ > 5)
```

Esto crea un subconjunto de 73 observaciones de nuestros datos originales. Tenga en cuenta que las observaciones con un *desaparecido* (N / A) valor de **democ** no se incluirá en el subconjunto. Las observaciones faltantes también se excluirían si hiciéramos una declaración mayor o igual a 8.

Como ejemplo de selección de variables, si quisiéramos centrarnos solo en la democracia y la riqueza, podríamos mantener solo estas dos variables y un índice para todas las observaciones:

```
dem.wealth <- subset (hmnrghts, select = c (país, democ, gnpcats))
```

Un medio alternativo de seleccionar las variables que deseamos conservar es utilizar un signo menos después de la Selección y enumere solo las columnas que deseamos eliminar. Por ejemplo, si quisiéramos todas las variables excepto los dos indicadores de si un país está en guerra, podríamos escribir:

```
no.war <- subconjunto (hmnrghts, select = -c (civ_war, int_war))
```

Además, los usuarios tienen la opción de llamar tanto al subconjunto y Selección opciones si desean elegir un subconjunto de variables sobre un conjunto específico de observaciones.

«Esto contrasta con programas como Stata, que tratan los valores perdidos como infinitos positivos. En Stata, si se incluyen las observaciones faltantes depende del tipo de declaración booleana que se haga. R es más consistente en que los casos que faltan siempre se excluyen.

2.4.2 Recodificar variables

Un aspecto final de la limpieza de datos que surge a menudo es la necesidad de recodificar las variables. Esto puede surgir porque la forma funcional de un modelo requiere una transformación de una variable, como un logaritmo o un cuadrado. Alternativamente, algunos de los valores de los datos pueden ser engañosos y, por lo tanto, deben ser recodificados como faltantes u otro valor. Otra posibilidad más es que las variables de dos conjuntos de datos deban codificarse en la misma escala: por ejemplo, si un analista ajusta un modelo con datos de encuestas y luego hace pronósticos utilizando datos del censo, entonces las variables de la encuesta y del censo deben codificarse del mismo camino.

Para las transformaciones matemáticas de variables, la sintaxis es sencilla y sigue la forma del ejemplo siguiente. Supongamos que queremos la población real de cada país en lugar de su logaritmo:

```
hmnrghts $ pop <- exp (hmnrghts $ lpop)
```

Simplemente, estamos aplicando la función exponencial (Exp) a un valor registrado para recuperar el valor original. Sin embargo, cualquier tipo de operador matemático podría sustituirse por Exp. Una variable se puede elevar al cuadrado (2), registradoIniciar sesión()), tomar la raíz cuadrada(sqrt ()), etc. La suma, resta, multiplicación y división también son válidas, ya sea con un escalar de interés o con otra variable. Supongamos que quisiéramos crear una variable ordinal codificada con 2 si un país estaba en una guerra civil y en una guerra internacional, 1 si estuvo involucrado en cualquiera de ellas y 0 si no estuvo involucrado en ninguna guerra. Podríamos crear esto agregando las variables de guerra civil y guerra internacional:

```
hmnrghts $ war.ord <- hmnrghts $ civ_war + hmnrghts $ int_war
```

Sin embargo, una tabla rápida de nuestra nueva variable revela que ninguna nación tuvo ambos tipos de conflicto en 1993.

Otro problema común que se debe abordar es cuando los datos se presentan en un formato no deseado. Nuestra variable **gnpcats** en realidad está codificado como una variable de texto. Sin embargo, es posible que deseemos recodificar esto como una variable ordinal numérica. Hay dos formas de lograr esto. El primero, aunque toma varias líneas de código, se puede completar rápidamente con una buena cantidad de copiar y pegar:

```
hmnrghts $ gnp.ord <- NA
hmnrghts $ gnp.ord [hmnrghts $ gnpcats == "<1000"] <- 1 hmnrghts $
gnp.ord [hmnrghts $ gnpcats == "1000-1999"] <- 2 hmnrghts $ gnp.ord
[hmnrghts $ gnpcats == "2000-2999"] <- 3 hmnrghts $ gnp.ord
[hmnrghts $ gnpcats == "3000-3999"] <- 4 hmnrghts $ gnp.ord
[hmnrghts $ gnpcats == "> 4000"] <- 5
```

Aquí, se creó una variable en blanco, y luego los valores de la nueva variable se completaron dependiendo de los valores de la antigua usando declaraciones booleanas.

Una segunda opción para recodificar los datos GNP se puede lograr a través de John Fox's **Cacompañar a arespondido rla egresióncarro**) Biblioteca. Como biblioteca escrita por el usuario, debemos descargarla e instalarla antes del primer uso. La instalación de una biblioteca es sencilla. Primero, escriba:

```
install.packages ("coche")
```

Una vez que la biblioteca está instalada (nuevamente, un paso que no necesita repetirse a menos que R se reinstala), las siguientes líneas generarán nuestra medida del PNB per cápita recodificada:

```
biblioteca (coche)
hmnrghts $ gnp.ord.2 <-recode (hmnrghts $ gnpcats, "'<1000" = 1;
"1000-1999" = 2; "2000-2999" = 3; "3000-3999" = 4; "> 4000" = 5 ')
```

Tenga cuidado de que el recodificar el mando es delicado. Entre los apóstrofes, todas las reasignaciones de valores antiguos a nuevos se definen separadas por punto y coma. Un solo espacio entre los apóstrofes generará un error. A pesar de esto, recodificar puede ahorrar a los usuarios un tiempo considerable en la limpieza de datos. La sintaxis básica de recodificar, por supuesto, podría usarse para crear variables ficticias, variables ordinales o una variedad de otras variables recodificadas. Así que ahora dos métodos han creado una nueva variable, cada una codificada del 1 al 5, donde 5 representa el PNB per cápita más alto.

Otro tipo estándar de recodificación que podríamos querer hacer es crear una variable ficticia que se codifique como 1 si la observación cumple ciertas condiciones y 0 en caso contrario. Por ejemplo, supongamos que en lugar de tener categorías de PNB, solo queremos comparar la categoría más alta de PNB con todas las demás:

```
hmnrghts $ gnp.dummy <-as.numeric (hmnrghts $ gnpcats == "> 4000")
```

Al igual que con nuestro ejemplo anterior de encontrar democracias involucradas en una guerra civil, aquí usamos una declaración lógica y la modificamos con el `as.numeric` declaración, que convierte cada CIERTO en un 1 y cada uno FALSO en un 0.

Variables categóricas en R se le puede dar una designación especial como *factores*. Si designa una variable categórica como factor, R lo tratará como tal en la operación estadística y creará variables ficticias para cada nivel cuando se utilice en una regresión. Si importa una variable sin codificación numérica, R llamará automáticamente a la variable a *personaje* vector, y convierta el vector de caracteres en un factor en la mayoría de los comandos de análisis. Sin embargo, si lo preferimos, podemos designar que una variable es un factor de antemano y abrir una variedad de comandos útiles. Por ejemplo, podemos designar `país` como factor:

```
hmnrghts $ país <- as.factor (hmnrghts $ país) niveles (hmnrghts $
país)
```

Darse cuenta de R permite al usuario poner la misma cantidad (en este caso, la variable **país**) a ambos lados de un operador de asignación. Esta asignación recursiva toma los valores antiguos de una cantidad, hace que el lado derecho cambie y luego reemplaza los nuevos valores en el mismo lugar en la memoria. El comando `niveles` nos revela los diferentes valores registrados del factor.

Para cambiar qué nivel es el primer nivel (por ejemplo, para cambiar qué categoría R utilizará como categoría de referencia en una regresión) utilice el `relevel` comando. El siguiente código establece "estados unidos" como la categoría de referencia para **país**:

```
hmnrghts $ país <-relevel (hmnrghts $ país, "estados unidos") niveles (hmnrghts $
país)
```

Ahora, cuando vemos los niveles del factor, "estados unidos" aparece como el primer nivel, y el primer nivel es siempre nuestro grupo de referencia.

2.5 Fusionar y remodelar datos

Dos tareas finales que son comunes a la gestión de datos son fusionar conjuntos de datos y remodelar los datos del panel. Al considerar ejemplos de estas dos tareas, consideremos una actualización de Poe et al. (1999) datos: Específicamente, Gibney et al. (2013) han continuado codificando datos para la Escala de Terror Político. Usaremos las oleadas de 1994 y 1995 de los datos actualizados. Las variables en estas ondas son:

País: Una variable de carácter que enumera el país por nombre.

COWAlpha: Abreviatura de país de tres caracteres de Correlates of War

conjunto de datos.

VACA: Variable numérica de identificación de países del conjunto de datos Correlates of War.

Banco Mundial: Abreviatura de país de tres caracteres utilizada por el Banco Mundial.

Amnistía 1994 / Amnistía 1995: La escala de terror político de Amnistía Internacional.

Las puntuaciones van desde 1 (terrorismo de estado bajo, menor número de violaciones de la integridad personal) a 5 (mayor número de violaciones de la integridad personal).

Departamento de Estado 1994 / Departamento de Estado 1995: La escala de políticas del Departamento de Estado de EE.

terror. Las puntuaciones van desde 1 (terrorismo de estado bajo, menor número de violaciones de la integridad personal) a 5 (mayor número de violaciones de la integridad personal).

Para las dos últimas variables, el nombre de la variable depende de qué ola de datos se esté estudiando, y el sufijo indica el año. Tenga en cuenta que estos datos tienen cuatro variables de identificación: está diseñado explícitamente para facilitar el uso de estos datos por parte de los investigadores. Cada índice facilita que un investigador vincule estas medidas de terror político con la información proporcionada por el Banco Mundial o el conjunto de datos de Correlates of War. Esto debería mostrar cuán omnipresente es el acto de fusionar datos para la investigación de Ciencias Políticas.

Con este fin, practiquemos *fusionando datos*. En general, la combinación de datos es útil cuando el analista tiene dos marcos de datos separados que contienen información sobre las mismas observaciones. Por ejemplo, si un politólogo tuviera un marco de datos con datos económicos por país y un segundo marco de datos que contenga los resultados electorales por país, el académico podría querer fusionar los dos conjuntos de datos para vincular los factores económicos y políticos dentro de cada país. En nuestro caso, supongamos que simplemente quisiéramos vincular el puntaje de terror político de cada país de 1994 a su puntaje de terror político de 1995. Primero, descargue los conjuntos de datos `pts1994.csv` y `pts1995.csv` del Dataverse en la página vii o el enlace del contenido del capítulo en la página 13. Como antes, es posible que deba usar `setwd` apuntar `R` a la carpeta donde ha guardado los datos. Después de esto, ejecute el siguiente código para cargar los datos relevantes:

```
hmnrghts.94 <-read.csv ("pts1994.csv")
hmnrghts.95 <-read.csv ("pts1995.csv")
```

Estos datos están separados por comas, por lo que `read.csv` es el mejor comando en este caso.

Si quisiéramos echar un vistazo a las primeras observaciones de nuestra onda de 1994, podríamos escribir `cabeza (hmnrghts.94)`. Esto imprimirá lo siguiente:

País COWAlpha COW WorldBank				
1	Estados Unidos	EE.UU	2	EE.UU
2	Canadá	LATA	20	LATA
3	Bahamas	BHM	31	BHS
4	Cuba	CACHORRO	40	CACHORRO
5	Haití	HAI	41	HTI
6	República Dominicana	DOM	42	DOM
Amnistía 1994 Departamento de Estado 1994				
1	1	N / A		
2	1	1		
3	1	2		
4	3	3		
5	5	4		
6	2	2		

Similar, [cabeza \(hmnrghts.95\)](#) imprimirá las primeras observaciones de nuestro 1995
onda:

País COWAlpha COW WorldBank				
1	Estados Unidos	EE.UU	2	EE.UU
2	Canadá	LATA	20	LATA
3	Bahamas	BHM	31	BHS
4	Cuba	CACHORRO	40	CACHORRO
5	Haití	HAI	41	HTI
6	República Dominicana	DOM	42	DOM
Amnistía 1995 Departamento de Estado 1995				
1	1	N / A		
2	N / A	1		
3	1	1		
4	4	3		
5	2	3		
6	2	2		

Como podemos ver en nuestra mirada a la parte superior de cada marco de datos, los datos están ordenados de manera similar en cada caso, y nuestras cuatro variables de índice tienen el mismo nombre en cada conjunto de datos respectivo. Solo necesitamos un índice para fusionar nuestros datos y los otros tres son redundantes. Por esta razón, podemos eliminar tres de las variables de índice de los datos de 1995:

```
hmnrghts.95 <- subconjunto(hmnrghts.95,  
  select = c(COW, Amnistía 1995, Departamento de Estado 1995))
```

Optamos por eliminar los tres índices de texto para fusionarlos en un índice numérico. Esta elección es arbitraria, sin embargo, porque R tampoco tiene problemas para fusionar una variable de carácter. (Intente replicar este ejercicio fusionando COWAlpha, por ejemplo.)

Para combinar nuestros datos de 1994 y 1995, pasamos ahora a la unir mando.⁹ Escribimos:

```
hmnrghts.wide <-merge(x = hmnrghts.94, y = hmnrghts.95, by = c("VACA"))
```

Dentro de este comando, la opción X se refiere a un conjunto de datos, mientras que y es el otro. Al lado de la opción, nombramos una variable de identificación que identifica de forma única cada observación. El comando en realidad permite a los usuarios nombrar múltiples variables si se necesitan varias para identificar de forma única cada observación: por ejemplo, si un investigador estaba fusionando datos donde la unidad de análisis era un país-año, una variable de país y una variable de año podrían ser esenciales para cada fila de forma única. En tal caso, la sintaxis podría leer, `by = c("VACA", "año")`. Como otra opción más, si los dos conjuntos de datos tuvieran el mismo índice, pero las variables se nombraron de manera diferente, R permite una sintaxis como, `by.x = c("VACA")`, `by.y = c("cowCode")`, lo que transmite que las variables de índice con nombres diferentes son el nombre.

Una vez que hayamos combinado nuestros datos, podemos obtener una vista previa del producto terminado escribiendo `cabeza(hmnrghts.wide)`. Esto imprime:

VACA		País COWAlpha WorldBank Amnistía 1994		
1	2	Estados Unidos	EE.UU	1
2	20	Canadá	LATA	1
3	31	Bahamas	BHM	1
4	40	Cuba	CACHORRO	3
5	41	Haití	HAI	5
6	42	República Dominicana	DOM	2

Departamento de Estado 1994 Amnistía 1995		Departamento de Estado 1995	
1	N / A	1	N / A
2	1	N / A	1
3	2	1	1
4	3	4	3
5	4	2	3
6	2	2	2

Como podemos ver, las puntuaciones de 1994 y 1995 para Amnistía y Departamento de Estado se registran en un lugar para cada país. Por lo tanto, nuestra fusión fue exitosa. Por defecto, R excluye cualquier observación de cualquiera de los conjuntos de datos que no tenga un *vinculado* observación (por ejemplo, valor equivalente) del otro conjunto de datos. Entonces, si usa los valores predeterminados y el nuevo conjunto de datos incluye el mismo número de filas que los dos conjuntos de datos anteriores, todas las observaciones se vincularon e incluyeron. Por ejemplo, podríamos escribir:

```
tenue(hmnrghts.94); tenue(hmnrghts.95); tenue(hmnrghts.wide)
```

Esto nos diría rápidamente que tenemos 179 observaciones en ambas entradas, así como en el conjunto de datos de salida, lo que muestra que no perdimos ninguna observación. Otras opciones dentro de `unir` están `all.x`, `all.y`, y `todas`, que le permiten especificar si se debe forzar

⁹junto al unir, la dplyr El paquete ofrece varios comandos de unión de datos que también puede resultarle útil, según sus necesidades.

la inclusión de todas las observaciones del conjunto de datos X, el conjunto de datos y, y de cualquier conjunto de datos, respectivamente. En este caso,R codificaría N / A valores para observaciones que no tenían un caso vinculado en el otro conjunto de datos.

Como punto final de la gestión de datos, a veces necesitamos *remodelar* nuestros datos. En el caso de nuestro conjunto de datos fusionado,hmnrghs.wide, Hemos creado un conjunto de datos de panel (por ejemplo, un conjunto de datos que consta de observaciones repetidas de los mismos individuos) que se encuentra en *formato amplio*. Formato amplio significa que cada fila de nuestros datos define un individuo de estudio (un país) mientras que nuestras observaciones repetidas se almacenan en variables separadas (p. Ej., Amnistía 1994 y Amnistía 1995 récord de puntuaciones de Amnistía Internacional durante dos años distintos). En la mayoría de los modelos de datos de panel, necesitamos que nuestros datos estén en*formato largo*, o formato apilado. El formato largo significa que necesitamos dos variables de índice para identificar cada fila, una para el individuo (por ejemplo, país) y otra para el momento de la observación (por ejemplo, año). Mientras tanto, cada variable (p. Ej.,Amnistía) solo usará una columna. R nos permite remodelar nuestros datos de ancho a largo, o de largo a ancho. Por lo tanto, sea cual sea el formato de nuestros datos, podemos adaptarlos a nuestras necesidades.

Para remodelar nuestros datos de terror político de formato amplio a largo, utilizamos el remodelar **mando:**

```
hmnrghs.long <-reshape(hmnrghs.wide, varying = c ("Amnistía.1994",
"StateDept.1994", "Amnistía.1995", "StateDept.1995"), timevar = "año",
idvar = "VACA", dirección = "largo", sep = ".")
```

Dentro del comando, el primer argumento es el nombre del marco de datos que deseamos remodelar. Lavariar término enumera todas las variables que representan observaciones repetidas a lo largo del tiempo. *Consejo:* Asegúrese de que las observaciones repetidas de la misma variable tengan el mismo *prefijo* nombre (p. ej., Amnistía o Departamento de Estado) y luego el *sufijo* (p.ej, 1994 o 1995) informa constantemente el tiempo. Latimevar término nos permite especificar el nombre de nuestro nuevo índice de tiempo, al que llamamos año. La idvar término enumera la variable que identifica de forma única a los individuos (países, en nuestro caso). Con dirección especificamos que queremos convertir nuestros datos en largo formato. Por último, elsep ofertas de comando R una pista de qué carácter separa nuestros prefijos y sufijos en las variables de observación repetidas: Dado que un punto (.) separa estos términos en cada uno de nuestros Amnistía y Departamento de Estado variables, lo denotamos aquí.

Una vista previa del resultado puede ser visto escribiendo **cabeza**. Esto

huellas dactilares:

	VACA	País	COWAlpha WorldBank	año
2.1994	2	Estados Unidos	EE.UU	Estados Unidos 1994
20.1994	20	Canadá	LATA	CAN 1994
31.1994	31	Bahamas	BHM	BHS 1994
40.1994	40	Cuba	CACHORRO	CUB 1994
41.1994	41	Haití	HAI	HTI 1994
42.1994	42	República Dominicana	DOM	DOM 1994
		Departamento de Estado de Amnistía		
2.1994	1	NA		
20.1994	1	1		

31.1994	1	2
40.1994	3	3
41.1994	5	4
42.1994	2	2

Note que nosotros ahora solo tengo una variable para Amnistía y uno para StateDept. Ahora tenemos una nueva variable llamada año, así que entre VACA y año, cada fila identifica unívocamente cada país-año. Dado que los datos están ordenados de forma natural, la parte superior de nuestros datos solo muestra observaciones de 1994. Mecanografía `cabeza(hmnrghs.long [hmnrghs.long $ año == 1995,])` muestra Veamos las primeras observaciones de 1995:

VACA		País COWAlpha WorldBank año		
2.1995	2	Estados Unidos	EE.UU	Estados Unidos 1995
20.1995	20	Canadá	LATA	CAN 1995
31.1995	31	Bahamas	BHM	BHS 1995
40.1995	40	Cuba	CACHORRO	CUB 1995
41.1995	41	Haití	HAI	HTI 1995
42.1995	42	República Dominicana	DOM	DOM 1995
Departamento de Estado de Amnistía				
2.1995	1	N / A		
20.1995	N / A	1		
31.1995	1	1		
40.1995	4	3		
41.1995	2	3		
42.1995	2	2		

Como podemos ver, toda la información se conserva, ahora en formato largo (o apilado). Como ilustración final, suponga que hemos comenzado con un conjunto de datos que estaba en formato largo y queríamos uno en formato ancho. Para probar esto, reformaremos `hmnrghs.long` e intentar recrear nuestro amplio conjunto de datos original. Para hacer esto, escribimos:

```
hmnrghs.wide.2 <- reshape(hmnrghs.long,
  v.names = c("Amnistía", "Departamento de Estado"), timevar = "año", idvar
  = "VACA", dirección = "ancho", sep = ".")
```

Algunas opciones ahora han cambiado: ahora usamos el `v.nombres` comando para indicar las variables que incluyen observaciones repetidas. `Latimevar` El parámetro ahora debe ser una variable dentro del conjunto de datos, al igual que `idvar` es decir, para separar a los individuos de puntos de tiempo repetidos. `Nuestradirección` el término es ahora amplio porque queremos convertir estos datos en formato ancho. Por último, `elsep` comando especifica el carácter que R utilizará para separar los prefijos de los sufijos en la forma final. Escribiendo `cabeza(hmnrghs.wide.2)` en la consola, ahora verá que este nuevo conjunto de datos recrea el conjunto de datos ancho original.

Este capítulo ha cubierto la variedad de medios de importar y exportar datos en R. También ha analizado cuestiones de gestión de datos como valores perdidos, subconjuntos, recodificación de datos, fusión de datos y remodelación de datos. Con la capacidad de limpiar y administrar datos, ahora estamos listos para comenzar a analizar nuestros datos. A continuación, procedemos a la visualización de datos.

2.6 Problemas de práctica

Como conjunto de datos de práctica, descargaremos y abriremos un subconjunto del Estudio Electoral Nacional Estadounidense de 2004 utilizado por Hanmer y Kalkan (2013). Este conjunto de datos se llama `hanmerKalkanANES.dta`, y está disponible en el Dataverse al que se hace referencia en la página vii o en el enlace de contenido del capítulo en la página 13. Estos datos están en formato Stata, así que asegúrese de cargar la biblioteca correcta y use el comando correcto al abrir. (*Insinuación:*

Cuando utilice el comando adecuado, asegúrese de especificar el `convert.factors = F` opción dentro de él para obtener un resultado más fácil de leer.) Todas las variables en este conjunto de datos se relacionan con las elecciones presidenciales de EE. UU. de 2004, y son: un número de identificación del encuestado (

Identificación del caso), evaluaciones económicas retrospectivas (**retecon**), evaluación de George

El manejo de W. Bush de la guerra en Irak (**bushiraq**), un indicador de si el encuestado votó por Bush (**voto previo**), partidismo en una escala de siete puntos (**partyid**), ideología en una escala de siete puntos (**ideol7b**), un indicador de si el encuestado es blanco (**blanco**), un indicador de si el encuestado es mujer (**mujer**), edad del encuestado (**edad**), nivel de educación en una escala de siete puntos (**educ1_7**), e ingresos en una escala de 23 puntos (**ingreso**). (La variable **exptrnout2** puede ser ignorado.)

1. Una vez que haya cargado los datos, haga lo siguiente para verificar su trabajo:

- (a) Si preguntas R para devolver los nombres de las variables, ¿qué dice la lista? ¿Es correcto?
- (b) Usando el comando `head`, ¿cómo se ven las primeras líneas?
- (c) Si usa el comando `describe`, ¿los datos parecen una hoja de cálculo adecuada?

2. Utilice el resumen comando en todo el conjunto de datos. ¿Qué puedes aprender de inmediato? ¿Cuántas observaciones faltantes tienes?

3. Intente crear subconjuntos de datos de varias formas:

- (a) Cree una copia del conjunto de datos que elimine todas las observaciones faltantes con eliminación por lista. ¿Cuántas observaciones quedan en esta versión?
- (b) Cree una segunda copia que solo incluya el número de identificación del encuestado, evaluaciones económicas retrospectivas y evaluación del manejo de Bush en Irak.

4. Cree algunas variables nuevas:

- (a) La escala de partidismo de siete puntos (**partyid**) se codifica de la siguiente manera: 0 = Demócrata fuerte, 1 = Demócrata débil, 2 = Demócrata inclinado independiente, 3 = Independiente no inclinado, 4 = Republicano inclinado independiente, 5 = Republicano débil y 6 = Republicano fuerte. Cree dos nuevas variables de indicador. El primero debe codificarse 1 si la persona se identifica como demócrata de alguna manera (incluidos los independientes que se inclinan por los demócratas) y 0 en caso contrario. La segunda variable nueva debe codificarse 1 si la persona se identifica como republicana de alguna manera (incluidos los independientes que se inclinan por la republicana) y 0 en caso contrario. Para cada una de estas dos nuevas variables, ¿qué resumen comando volver por ellos?
- (b) Cree una nueva variable que sea el valor al cuadrado de la edad del encuestado en años. Lo que hace el resumen comando para esta nueva variable?
- (c) Cree una nueva versión de la variable de ingreso que tenga solo cuatro categorías. La primera categoría debe incluir todos los valores de **ingreso** que el rango de 1 a 12,

el segundo del 13 al 17, el tercero del 18 al 20 y el último del 21 al 23.

Utilizar el `mesa` comando para ver la frecuencia de cada categoría.

- (d) Bonificación: utilice el `mesa` comando para comparar la versión de ingresos de 23 categorías con la versión de ingresos de cuatro categorías. ¿Codificó la nueva versión correctamente?