

Programming Historian en español



Datos tabulares en R (/es/lecciones/datos-tabulares-en-r).

Taryn Dewar

Esta lección te enseña una forma rápida de analizar grandes cantidades de datos en forma de tabla, haciendo la investigación más rápida y efectiva.

👤 Revisado por pares (<https://github.com/programminghistorian/ph-submissions/issues/164>).

📄 CC-BY 4.0 (<https://creativecommons.org/licenses/by/4.0/deed.en>).

💰 Apoyar PH (</es/apoyanos#donaciones>).


editado por

- Adam Crymble

revisado por

- James Baker  (<https://orcid.org/0000-0002-2682-6922>)
- John Russell

traducido por

- Jennifer Isasi  (<https://orcid.org/0000-0002-4295-895X>)

traducción editada por

- Antonio Rojas Castro  (<https://orcid.org/0000-0002-8916-4997>).

traducción revisada por

- Joseba Moreno
- Antonio Rojas Castro  (<https://orcid.org/0000-0002-8916-4997>).

publicado

| 2016-09-05

traducido


| 2018-08-20

modificado

| 2022-03-15

dificultad

| Baja

 <https://doi.org/10.46430/phes0034>

Disponible en: [EN \(/en/lessons/r-basics-with-tabular-data\)](https://en/lessons/r-basics-with-tabular-data) (original) | [ES \(/es/lecciones/datos-tabulares-en-r\)](https://es/lecciones/datos-tabulares-en-r) | [PT \(/pt/liceos/nocoes-basicas-R-dados-tabulares\)](https://pt/liceos/nocoes-basicas-R-dados-tabulares)

Objetivos

A medida que se digitalizan más y más datos históricos, contar con una forma rápida de analizar grandes cantidades de datos en forma de tabla hace la investigación más rápida y efectiva.

[R \(https://es.wikipedia.org/wiki/R_\(lenguaje_de_programación\)\)](https://es.wikipedia.org/wiki/R_(lenguaje_de_programación)), es un lenguaje de programación para el análisis estadístico. Como tal, puede ser usado para completar análisis cuantitativos en recursos históricos que incluyen test estadísticos, entre otros. Puesto que puedes ejecutar el mismo código repetidas veces sobre las mismas fuentes, R permite analizar datos de forma rápida y produce resultados reproducibles. Al poder guardar tu código, R te permite reusar o revisar funciones para futuros proyectos, haciéndose una parte flexible de tus herramientas.

Este tutorial no presupone conocimiento previo de R. Te guiará por algunas de las funciones básicas de R, sirviendo como una introducción al lenguaje. Te guiará en el proceso de instalación, te explicará algunas de las herramientas que puedes utilizar en R, y te explicará cómo trabajar con grupos de datos mientras investigas. El tutorial hará esto a través de mini-lecciones que te enseñarán el tipo de recursos con los que R trabaja bien y ejemplos de cómo hacer cálculos para encontrar información que pueda ser relevante a la investigación histórica. La lección también cubrirá diferentes métodos de entrada en R, como las matrices y el uso de archivos CSV (valores separados por comas).

¿Para quién es útil?

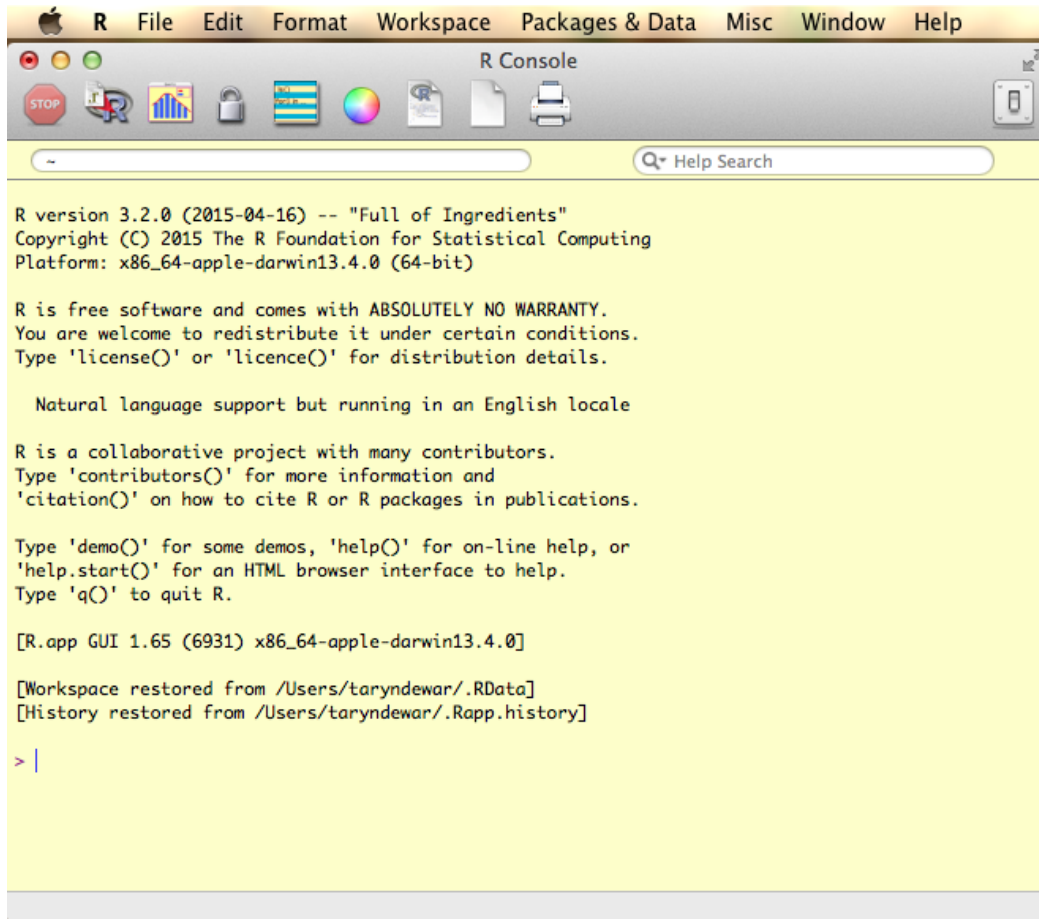
R es ideal para analizar grandes cantidades de datos que llevarían demasiado tiempo de computación manual. Una vez que entiendas cómo escribir algunas de las funciones básicas y a importar tus propios archivos de datos, podrás analizar y visualizar los datos de forma rápida y eficiente.

Mientras que R es una gran herramienta para datos tabulares, puede que encuentres más útiles otros acercamientos al análisis de fuentes no tabulares (como transcripciones de periódicos). Si te interesa estudiar este tipo de recursos, echa un vistazo al resto de lecciones en [The Programming Historian en español \(/es\)](https://es/lecciones/datos-tabulares-en-r).

Instalar R

R es un lenguaje de programación y un entorno para trabajar con datos. Se puede usar en la consola de R en la [línea de comandos \(/es/lecciones/introduccion-a-bash\)](https://es/lecciones/introduccion-a-bash) o en el [Interfaz de R \(https://www.rstudio.com/products/rstudio/download/\)](https://www.rstudio.com/products/rstudio/download/), ambos disponibles solamente en inglés. Este tutorial se centrará en el uso de la consola de R. Para empezar con R, descarga el programa desde [The Comprehensive R Archive Network \(https://cran.r-project.org/\)](https://cran.r-project.org/). R es compatible con Linux, Mac y Windows.

Cuando inicies la consola de R por primera vez, se abrirá una ventana parecida a esta:



La consola de R en Mac.

Usar la consola de R

La consola de R es un buen lugar para empezar si eres nuevo con R porque está específicamente diseñada para este lenguaje y tiene funciones que son específicas a él.

La consola es donde escribirás los comandos. Para limpiar la pantalla inicial, ve a 'Edit' (editar) en la barra de menús y selecciona 'Clear Console' (limpiar consola). Esto te proporcionará una página limpia. También puedes cambiar la apariencia de la consola clicando en la rueda de colores en la parte superior de la consola en un Mac, o seleccionando 'GUI Preferences' (preferencias de la Interfaz Gráfica de Usuario) en el menú 'Edit' en un PC. Puedes ajustar el color del fondo de pantalla y también el color de la fuente para tus funciones.

Usar grupos de datos

Antes de trabajar con tus propios datos, ayuda que te hagas una idea del funcionamiento de R usando los grupos de datos que éste incluye. Puedes buscar en los grupos de datos ejecutando `data()` en la consola. Esto mostrará una lista de todos los grupos de datos disponibles en una ventana aparte; la lista incluye los títulos de todos los grupos de datos además de una pequeña descripción de la información de cada uno.

Primero, necesitas cargar el dataset `AirPassengers`¹ en la sesión de R. Escribe `data(AirPassengers)` y pulsa *Intro*. Para ver el dataset, escribe `AirPassengers` en la siguiente línea de código y pulsa *Intro*. Esto imprimirá o mostrará una tabla con el número de pasajeros que voló en aerolíneas internacionales entre enero de 1949 y diciembre de 1960, en miles. Deberías poder ver:

```
> data(AirPassengers)
> AirPassengers
  Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Ahora puedes usar R para responder a un número de preguntas basadas en estos datos. Por ejemplo, ¿cuáles fueron los meses más populares para volar? ¿Hubo un incremento en viajes internacionales con el tiempo? Probablemente puedas encontrar la respuesta a estas preguntas simplemente escaneando esta tabla pero no tan rápido como lo hace el ordenador. ¿Y qué ocurre si tenemos muchos más datos?

Funciones básicas

Se puede usar R para calcular un número de valores que pueden ser útiles mientras investigas un grupo de datos. Por ejemplo, puedes encontrar la media ([https://es.wikipedia.org/wiki/Media_\(matemáticas\)](https://es.wikipedia.org/wiki/Media_(matemáticas))), la mediana ([https://es.wikipedia.org/wiki/Mediana_\(estadística\)](https://es.wikipedia.org/wiki/Mediana_(estadística))), y los valores mínimos y máximos en el conjunto de datos. Para obtener la media y la mediana en el conjunto de datos, puedes ejecutar `mean(AirPassengers)` y `median(AirPassengers)` en la consola respectivamente. ¿Qué ocurre si quieres calcular más de un único valor al mismo tiempo? Para producir un resumen de los datos, ejecuta `summary(AirPassengers)` (resumen) en la consola. Esto te dará los puntos mínimo y máximo del conjunto, así como la media, la mediana y los valores cuartiles (<https://es.wikipedia.org/wiki/Cuartil>), primero y tercero.

```
> summary(AirPassengers)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 104.0   180.0   265.5   280.3   360.5   622.0
```

Al crear un resumen podemos ver que el número mínimo de pasajeros entre enero de 1949 y diciembre de 1960 fue de 104.000, y que el número máximo de pasajeros fue de 622.000. La media muestra que aproximadamente 280.300 personas viajaron por cada mes que se recogieron estos datos. Estos valores pueden ser útiles para ver el grado de variación en número de pasajeros en el tiempo.

Usar la función `summary()` es una forma de obtener una vista general del conjunto de datos pero ¿qué ocurre si te interesa un subconjunto de datos como un año particular o algunos meses? En R puedes seleccionar diferentes puntos de datos (como un mes concreto) y un rango (como un año concreto) para calcular diferentes valores. Por ejemplo, puedes sumar el número de pasajeros de dos meses para determinar el total de pasajeros durante ese periodo de tiempo.

Intenta sumar los dos primeros valores del conjunto `AirPassengers` en la consola y luego pulsa 'Intro'. Deberías ver dos líneas que apuntan:

```
> 112+118
[1] 230
```

Esto te da el número total de pasajeros (en cientos de miles) que volaron en enero y febrero de 1949.

R puede hacer más que simple aritmética. Puedes crear objetos (https://es.wikipedia.org/wiki/Objeto_matemático) o variables ([https://es.wikipedia.org/wiki/Variable_\(programación\)](https://es.wikipedia.org/wiki/Variable_(programación))) para representar números y expresiones ([https://es.wikipedia.org/wiki/Expresión_\(informática\)](https://es.wikipedia.org/wiki/Expresión_(informática))). Por ejemplo, puedes dar el nombre `Ene1949` a la variable para el valor de enero de 1949. Escribe `Ene1949 <- 112` en la consola y luego `Ene1949`. La anotación `<-` asigna el valor `112` a la variable `Ene1949`. Deberías ver:

```
> Ene1949 <- 112
> Ene1949
[1] 112
```

R es sensible a minúsculas y mayúsculas, por tanto, has de tener cuidado al usar las mismas anotaciones cuando usas las variables que has asignado (o nombrado) en otras acciones. Sin embargo, el nombre de las variables puede ir en inglés o español, o cualquier otro idioma. Puedes leer el artículo de Rasmus Bååth, The State of Naming Conventions in R (https://journal.r-project.org/archive/2012-2/RJournal_2012-2_Baaath.pdf) (en inglés), para tener más información sobre la mejor forma de llamar a las variables.

Para eliminar una variable de la consola, escribe `rm()` (*remove* o borrar) con la variable de la que te quieras deshacer dentro de los paréntesis y pulsa 'Intro'. Para ver todas las variables que has designado escribe `ls()` (*list objects*) en la consola y pulsa 'Intro'; esto te ayudará a evitar la repetición de nombres en múltiples variables. Esto también es importante porque R guarda en su memoria todos los objetos que creas y aunque tú no puedas ver la variable llamada `x` en la consola, puede que la hayas creado antes y accidentalmente podrías sobrescribirla asignando el mismo nombre a otra variable.

Aquí está la lista de variables que hemos creado de momento:

```
> ls()
[1] "AirPassengers" "Ene1949"
```

Tenemos la variable `AirPassengers` y la variable `Ene1949`. Si borramos la variable `Ene1949` y reescribimos `ls()`, veremos:

```
> rm(Ene1949)
> ls()
[1] "AirPassengers"
```

Si en algún momento te atascas con una función o no puedes arreglar un error, escribe `help()` en la consola para abrir la página de ayuda. También puedes encontrar ayuda general usando el menú 'Help' en la parte de arriba de la consola. Si quieres cambiar algo en el código que ya has escrito, puedes reescribir el código en una nueva línea. Para ahorrar tiempo, también puedes usar los cursores en el teclado para desplazarte arriba y abajo en la consola y así encontrar la línea de código que quieres cambiar.

Puedes usar letras como variables pero cuando empieces a trabajar con tus propios datos es más fácil que asignes nombres que representen mejor los datos. Incluso con los datos de `AirPassengers()`, asignar variables que corresponden a meses o años específicos hace más fácil saber con qué puntos estás trabajando.

Práctica

A. Asigna variables para los puntos de datos de `AirPassengers()` de enero de 1950 y enero de 1960. Suma ambas variables en la siguiente línea.

B. Utiliza las variables que acabas de crear para encontrar diferencias entre los viajeros de 1950 y 1960.

Soluciones

A. Asigna variables para los puntos de datos de `AirPassengers()` de enero de 1950 y enero de 1960. Suma ambas variables en la siguiente línea.

```
> Ene1950 <- 115
> Ene1960 <- 417
> Ene1950+Ene1960
[1] 532
```

Esto significa que 532.000 personas viajaron en vuelos internacionales en enero de 1950 y de 1960.

B. Utiliza las variables que acabas de crear para encontrar diferencias entre los viajeros de 1950 y 1960.

```
> Ene1960-Ene1950
[1] 302
```

Esto significa que hubo 302.000 pasajeros más en los vuelos internacionales en enero de 1960 respecto a enero de 1950.

Establecer variables para puntos de datos puede ser tedioso, especialmente si los nombres que das son largos. Sin embargo, el proceso para asignar un rango de valores a una variable que contenga todos los datos de un año es similar. Hacemos esto creando listas llamadas 'vectores' usando el comando `c`. `c` aquí quiere decir 'combinar' y permite unir números en una variable común. Por ejemplo, puedes crear un vector para los datos de `AirPassengers` de 1949 y llamarlo `Air49`:

```
> Air49<- c(112,118,132,129,121,135,148,148,136,119,104,118)
```

Cada punto es accesible usando el nombre de la variable y su posición de indexado (empezando en 1). En este caso, `Air49[2]` contiene el valor que corresponde a febrero de 1949 - 118.

```
> Air49[2]
[1] 118
```

Puedes crear una lista de valores consecutivos usando dos puntos. Por ejemplo:

```
> y <- 1:10
> y
[1] 1 2 3 4 5 6 7 8 9 10
```

Con esto, puedes usar la siguiente expresión para definir una variable para los datos de `AirPassengers` de 1949.

```
> Air49 <- AirPassengers[1:12]
> Air49
[1] 112 118 132 129 121 135 148 148 136 119 104 118
```

`Air49[2]` ha seleccionado los doce primeros objetos en el conjunto de datos de `AirPassengers`. Esto te ofrece los mismos resultados que los de arriba, pero toma menos tiempo y además reduces la posibilidad de que un valor sea transcrito incorrectamente.

Para obtener el número total de pasajeros de 1949, puedes sumar todos los objetos del vector usando la función `sum()` (sumar).

```
> sum(Air49)
[1] 1520
```

Por tanto, el número total de pasajeros de 1949 fue de aproximadamente 1.520.000.

Finalmente, la función `length()` (longitud o tamaño) hace posible saber el número de objetos en un vector:

```
> length(Air49)
[1] 12
```

Práctica

1. Crea una variable para los datos de 1950 `AirPassengers`.
2. Imprime el segundo objeto en la serie de 1950.
3. ¿Cuál es el tamaño (*length*) de la secuencia en la pregunta 2?
4. ¿Cuántos pasajeros volaron en total en 1950?

Soluciones

1.

```
> Air50 <- AirPassengers[13:24]
Air50
[1] 115 126 141 135 125 149 170 170 158 133 114 140
```

2.

```
> Air50[2]
[1] 126
```

3.

```
> length(Air50)
[1] 12
```

4.

```
> sum(Air50)
[1] 1676
```

Si quieres crear variables para todos los años en este conjunto de datos, puedes utilizar algunas de las herramientas que hemos estudiado para determinar el número de pasajeros de avión a lo largo del tiempo. Aquí tienes una lista de variables de 1949 a 1960, seguido del número total de pasajeros por año:

```
> Air49 <- AirPassengers[1:12]
Air50 <- AirPassengers[13:24]
Air51 <- AirPassengers[25:36]
Air52 <- AirPassengers[37:48]
Air53 <- AirPassengers[49:60]
Air54 <- AirPassengers[61:72]
Air55 <- AirPassengers[73:84]
Air56 <- AirPassengers[85:96]
Air57 <- AirPassengers[97:108]
Air58 <- AirPassengers[109:120]
Air59 <- AirPassengers[121:132]
Air60 <- AirPassengers[133:144]
```

```
> sum(Air49)
[1] 1520
sum(Air50)
[1] 1676
sum(Air51)
[1] 2042
sum(Air52)
[1] 2364
sum(Air53)
[1] 2700
sum(Air54)
[1] 2867
sum(Air55)
[1] 3408
sum(Air56)
[1] 3939
sum(Air57)
[1] 4421
sum(Air58)
[1] 4572
sum(Air59)
[1] 5140
sum(Air60)
[1] 5714
```

Según esta información, puedes ver que el número de pasajeros aumentó cada año. Con estos datos podrías ir más allá y determinar si hubo un incremento en el interés de vacacionar en ciertas épocas del año, o incluso el porcentaje de aumento de pasajeros a lo largo del tiempo.

Trabajar con bases de datos más grandes

Ten en cuenta que el ejemplo de más arriba no se adapta bien para bases de datos más grandes: contar los puntos de datos para encontrar los correctos sería muy tedioso. Piensa qué ocurriría si estuvieras buscando la información del año 96 en un conjunto de datos con 150 años de datos recogidos.

En realidad puedes seleccionar filas y columnas de datos específicos si el conjunto de datos está en un formato concreto. Carga los datos de `mtcars` en tu consola:

```
> data(mtcars)
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Este [conjunto de datos \(https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html\)](https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html), ofrece una perspectiva general de la *Motor Trend Car Road Tests* de la revista *Motor Trend* de 1974². Contiene información sobre cuántas millas por galón o kilómetros por litro podía viajar un coche³, el número de cilindros de motor de cada coche, los caballos, la relación del eje trasero, el peso y otras características de cada modelo. Los datos podrían ser utilizados para descubrir cuáles de estas características hicieron que cada tipo de coche fuera más o menos seguro para los pasajeros a lo largo del tiempo.

Puedes seleccionar las columnas introduciendo el nombre del conjunto de datos seguido de corchetes y el número de la fila o la columna de datos que te interese. Para ordenar las filas y las columnas, piensa en `dataset[x,y]`, siendo el `dataset` el conjunto con el que estás trabajando, la `x` la fila y la `y` la columna.

Si te interesara la primera fila de información del conjunto `mtcars`, ejecutarías lo siguiente en tu consola:

```
> mtcars[1,]
      mpg cyl disp  hp drat   wt  qsec vs am gear carb
Mazda RX4   21   6  160  110  3.9 2.62 16.46  0  1    4    4
```

Para ver una columna de los datos, puedes ejecutar:

```
> mtcars[,2]
[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 4 4 4 4 8 8 8 4 4 4 8 6 8 4
```

Esto también mostraría todos los valores bajo la categoría `cyl` (cilindrada). La mayoría de los modelos de coche tienen un cilindrada de 4, 6 u 8. También puedes seleccionar puntos de datos individuales ejecutando valores para `x` (fila) e `y` (columna):

```
> mtcars[1,2]
[1] 6
```

Esto ofrece el valor de la primera fila en la segunda columna. Desde aquí, puedes crear un resumen de una fila o de una columna de datos sin tener que contar el número de objetos en cada conjunto de datos. Por ejemplo, escribiendo `summary(mtcars[,1])` en la consola y pulsando 'Intro' obtienes el resumen de la capacidad de millas por galón de los diferentes coches en el conjunto de datos `mtcars`:


```
> summary(mtcars[,1])
  Min. 1st Qu.  Median    Mean 3rd Qu.   Max.
 10.40  15.42   19.20   20.09  22.80   33.90
```

El resumen indica que el máximo de eficiencia de combustible era de 33.9 millas por galón o 54.5 kilómetros por 3.78 litros, del Toyota Corolla, y el coche menos eficiente era el Lincoln Continental, que solo conseguía hacer 10.4 millas por galón, es decir, 16.7 kilómetros por 3.78 litros. Podemos encontrar los coches que coinciden con los puntos de datos mirando de nuevo a la tabla. Es mucho más fácil encontrar un valor específico que tratar de realizar el cálculo en tu cabeza o buscar en la hoja de cálculo.

Matrices

Ahora que entiendes mejor cómo funcionan algunas de las funciones básicas de R, podemos ver formas de usar esas funciones con nuestros propios datos. Esto incluye la creación de [matrices](https://es.wikipedia.org/wiki/Matriz_(matemáticas)) ([https://es.wikipedia.org/wiki/Matriz_\(matemáticas\)](https://es.wikipedia.org/wiki/Matriz_(matemáticas))), usando conjunto de datos pequeños. El beneficio de saber cómo construir matrices en R es que si solo tienes unos pocos puntos de datos para trabajar puedes crear una matriz en vez de un CSV (archivo de valores separados por comas) que luego tendrías que importar. Una de las formas más sencillas de crear una matriz es crear al menos dos variables o vectores y después unirlos. Por ejemplo, mira estos datos del [Old Bailey](https://es.wikipedia.org/wiki/Old_Bailey) (https://es.wikipedia.org/wiki/Old_Bailey) (o Tribunal Penal Central de Inglaterra y Gales) en línea:

Statistics

<< Back to Search Form

Tabulating decade against offence category where the transcription matches 'sword' , between 1674 and 1800. Counting by offence.

* - Incomplete Decade

	Total	Breaking Peace	Killing	Theft	Violent Theft	Royal Offences	Sexual Offences	Deception	Miscellaneous	Damage to Property
Total	1051 100%	21 100%	374 35.59%	359 34.16%	238 22.65%	26 2.47%	10 0.95%	16 1.52%	5 0.48%	2 0.19%
1670s*	23 100%	2 4.35%	13 56.52%	2 8.7%	7 30.43%	0 0%	0 0%	0 0%	0 0%	0 0%
1680s*	141 100%	2 1.42%	87 61.7%	30 21.28%	20 14.18%	1 0.71%	1 0.71%	0 0%	0 0%	0 0%
1690s*	166 100%	0 0%	89 53.61%	38 22.89%	36 21.69%	1 0.6%	0 0%	1 0.6%	1 0.6%	0 0%
1700s*	36 100%	0 0%	19 52.78%	13 36.11%	3 8.33%	0 0%	0 0%	1 2.78%	0 0%	0 0%
1710s*	108 100%	2 1.85%	44 40.74%	33 30.56%	26 24.07%	3 2.78%	0 0%	0 0%	0 0%	0 0%
1720s	129 100%	3 2.33%	51 39.53%	38 29.46%	31 24.03%	1 0.78%	3 2.33%	1 0.78%	1 0.78%	0 0%
1730s	99 100%	3 3.03%	17 17.17%	53 53.54%	24 24.24%	0 0%	1 1.01%	0 0%	1 1.01%	0 0%
1740s	78 100%	0 0%	14 17.95%	29 37.18%	32 41.03%	2 2.56%	0 0%	1 1.28%	0 0%	0 0%
1750s	61 100%	0 0%	11 18.03%	24 34.43%	24 39.34%	1 1.64%	2 3.28%	2 3.28%	0 0%	0 0%
1760s	49 100%	2 4.08%	9 18.37%	28 57.14%	4 8.16%	0 0%	2 4.08%	2 4.08%	0 0%	2 4.08%
1770s	77 100%	2 2.6%	5 6.49%	38 49.35%	21 27.27%	4 5.19%	1 1.3%	4 5.19%	2 2.6%	0 0%
1780s	52 100%	5 9.62%	6 11.54%	19 36.54%	8 15.38%	10 19.23%	0 0%	0 0%	2 3.85%	0 0%
1790s	26 100%	1 3.85%	7 26.92%	13 50%	2 7.69%	3 11.54%	0 0%	0 0%	0 0%	0 0%
1800s	6 100%	0 0%	2 33.33%	4 66.67%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%

Export

Conjunto de datos del Old Bailey para casos criminales en cada década entre 1670 y 1800.

El Old Bailey contiene estadísticas e información sobre los casos criminales celebrados en el Tribunal Penal Central de Londres entre 1674 y 1913. Si quisiéramos ver el número total de robos y robos con violencia para las décadas de entre 1670 y 1710, pondríamos esos números en una matriz.

Para hacer esto, creamos las variables `Robos` y `RobosViolentos` usando el total de cada década como puntos de datos:

```
> Robos <- c(2,30,38,13)
RobosViolentos <- c(7,20,36,3)
```

Para crear una matriz podemos usar la función `cbind()` (*column bind* o unión de columnas). Esto une `Robos` y `RobosViolentos` en columnas, representadas como `Crimen` aquí:

```
> Robos <- c(2,30,38,13)
RobosViolentos <- c(7,20,36,3)
Crimen <- cbind(Robos,RobosViolentos)
Crimen
      Robos RobosViolentos
[1,]     2           7
[2,]    30          20
[3,]    38          36
[4,]    13           3
```

También puedes establecer una matriz usando `rbind()`. `rbind()` une los datos en filas (*row bind* o unión de filas). Observa la diferencia entre `Crimen` y `Crimen2`:

```
> Crimen2 <- rbind(Robos,RobosViolentos)
> Crimen2
      [,1] [,2] [,3] [,4]
Robos      2  30  38  13
RobosViolentos 7  20  36   3
```

La segunda matriz también puede ser creada usando la expresión `t(Crimen)` (transposición de matriz), que genera lo inverso a `Crimen`.

También puedes contruir una matriz usando `matrix()`. Esto te permite transformar una secuencia de números, como el número de robos y robos violentos cometidos, en una matriz si no has creado variables separadas para estos puntos de datos:

```
> matrix(c(2,30,3,4,7,20,36,3),nrow=2)
      [,1] [,2] [,3] [,4]
[1,]     2    3    7   36
[2,]    30    4   20    3

[2,]    30    4   20    3
> matrix(c(2,30,3,4,7,20,36,3),ncol=2)
      [,1] [,2]
[1,]     2    7
[2,]    30   20
[3,]     3   36
[4,]     4    3
```

La primera parte de la función es la lista de números. Después, puedes determinar cuántas filas (`nrow=`) (número de filas) o columnas (`ncol=`) (número de columnas) tendrá la matriz.

La función `apply()` (aplicar) te permite realizar la misma función en cada fila o columna de una matriz. Hay tres partes en la función `apply`, en la que tienes que seleccionar: la matriz que estás utilizando, los términos que quieres usar y la función que quieres realizar sobre la matriz:

```
> Crimen
      Robos RobosViolentos
[1,]     2           7
[2,]    30          20
[3,]    38          36
[4,]    13           3
> apply(Crimen,1,mean)
[1]  4.5 25.0 37.0  8.0
```

Este ejemplo muestra la función `apply` usada en la matriz `Crimen` para calcular la media (*mean*) de cada fila y, por tanto, el número promedio de robos y de robos con violencia combinados que fueron cometidos en cada década. Si quieres saber la media de cada columna, usa un `2` en lugar de un `1` dentro de la función:

```
> apply(Crimen,2,mean)
      Robos RobosViolentos
20.75     16.50
```

Esto muestra el número promedio de robos y de robos con violencia entre las décadas.

Practica

1. Crea una matriz con dos columnas usando los siguientes datos de Quebrantamiento de la Paz y Asesinatos de entre 1710 y 1730 de la tabla de más arriba del Old Bailey: `c(2,3,3,44,51,17)`

2. Utiliza la función `cbind()` para unir `QuebrantamientoPaz <- c(2,3,3)` y `Asesinatos <- c(44,51,17)` .

3. Calcula la media de cada columna de la matriz usando la función `apply()` .

Soluciones

1.

```
> matrix(c(2,3,3,44,51,17),ncol=2)
      [,1] [,2]
[1,]    2  44
[2,]    3  51
[3,]    3  17
```

2.

```
> QuebrantamientoPaz <- c(2,3,3)
> Asesinatos <- c(44,51,17)
> PazAsesinatos <- cbind(QuebrantamientoPaz,Asesinatos)
> PazAsesinatos
      QuebrantamientoPaz Asesinatos
[1,]                2        44
[2,]                3        51
[3,]                3        17
```

3.

```
> apply(PazAsesinatos,2,mean)
> QuebrantamientoPaz      Asesinatos
>           2.666667      37.333333
```

Usar las matrices puede ser útil si estás trabajando con una cantidad pequeña de datos. Sin embargo, no siempre es la mejor opción porque las matrices pueden ser difíciles de leer. A veces es más fácil crear tu propio archivo usando un programa de hojas de cálculo como [Excel](https://es.wikipedia.org/wiki/Microsoft_Excel) (https://es.wikipedia.org/wiki/Microsoft_Excel) u [Open Office](https://www.openoffice.org/es/) (<https://www.openoffice.org/es/>). De esta manera te puedes asegurar de que toda la información que quieres estudiar está organizada. Además, puedes importar dicho archivo a R.

Cargar tu propio conjunto de datos en R

Ahora que has practicado con datos simples, estás preparado/a para trabajar con tus propios datos. Estos posiblemente están en una hoja de cálculo. ¿Cómo puedes trabajar con estos datos en R? Hay varias formas de hacer esto. La primera es cargar la hoja de cálculo directamente en R. Otra forma es importar un archivo CSV (valores separados por comas) o TXT (de texto) a R.

Para cargar un archivo de Excel directamente a la consola de R, primero tienes que instalar el paquete `readxl` (leer archivo excel). Para hacer esto, escribe `install.packages("readxl")` en la consola y pulsa Intro. Puede que tengas que comprobar que el paquete se ha instalado en la consola clicando la pestaña 'Packages&Data' (paquetes y datos) en el menú, seleccionando 'Package Manager' (gerente de paquetes) y después clicando en la caja junto al paquete `readxl` . Desde aquí, puedes seleccionar un archivo y cargarlo en R. Abajo tienes un ejemplo de lo que puede parecer cargar un archivo simple de Excel:

```
> x <- read_excel("Workbook2.xlsx")
> x
  a b
1 1 5
2 2 6
3 3 7
4 4 8
```

Después del comando `read_excel` está el nombre del archivo que has seleccionado. Los números de abajo corresponden a los datos en la hoja de cálculo que he usado de ejemplo. Ten en cuenta que las filas están numeradas y mis columnas están etiquetadas como lo están en la hoja de cálculo original.

Cuando cargas datos en R asegúrate de que el archivo al que estás accediendo esté en el directorio en que estás trabajando en tu ordenador. Para comprobar esto, puedes escribir `dir()` (directorio) o `getwd()` (mostrar ruta del directorio de trabajo) en la consola. Puedes cambiar el directorio si lo necesitas clicando en la pestaña de 'Miscellaneous' (miscelánea) en el menú de tu pantalla y seleccionando el directorio que deseas para R. Si no haces esto R no podrá encontrar el archivo correctamente.

Otra forma de cargar datos en R es usar un archivo CSV. Un archivo [CSV](https://es.wikipedia.org/wiki/Valores_separados_por_comas) (https://es.wikipedia.org/wiki/Valores_separados_por_comas) (valores separados por comas) muestra los valores en filas y columnas, separando éstas con comas. Puedes guardar cualquier documento creado en Excel como un archivo .csv y después cargarlo a R. Para usar un archivo CSV en R, asigna un nombre al archivo usando el

comando `<-` seguido de `read.csv(file="nombre-del-archivo.csv",header=TRUE,sep=",")` en la consola. `nombre-del-archivo` le indica a R qué archivo seleccionar mediante el comando `file=` (el archivo equivale a), mientras que la configuración del encabezado o `header=` como `TRUE` (verdadero) indica que la primera fila se trata del encabezado y no de variables. Con `sep` indicamos que hay una coma entre cada número y línea.

Normalmente, un CSV puede contener bastante información. Sin embargo, para comenzar, trata de crear un archivo CSV en Excel usando los datos del Old Bailey que hemos usado para las matrices. Establece las columnas para las fechas entre 1710 y 1730 además del número de crímenes de quebrantamientos de paz y de asesinatos para esas décadas. Guarda el archivo como "OldBailey.csv" e intenta cargarlo en R usando los pasos anteriores. Vas a ver:

```
> read.csv (file="OldBailey.csv", header=TRUE, sep=",")
Fecha QuebrantamientoPaz Asesinatos
1 1710                2      44
2 1720                3      51
3 1730                4      17
```

Ahora puedes acceder a los datos en R y realizar cálculos que te ayuden a estudiarlos. Los archivos CSV pueden ser más complejos que este ejemplo, así que también puedes abrir en R cualquier conjunto de datos en que estés trabajando.

Puedes importar archivos de texto (TXT) de una manera similar. Usa el comando `read.table()` para cargar archivos de texto en R, siguiendo la misma sintaxis que en el ejemplo de más arriba.

Guardar datos en R

Ahora que has cargado datos en R y conoces algunas formas de trabajar con los datos, ¿qué ocurre si quieres guardarlos en otro formato? La función `write.xlsx` te permite hacer precisamente eso: tomar datos de R y guardarlos en un archivo de Excel. Intenta guardar el archivo *Old Bailey* como un archivo de Excel. Primero tienes que cargar el paquete. Después, crea una variable para los datos de *Old Bailey* y crea el archivo:

```
> library(xlsx)
> write.xlsx(x= OldBailey, file= "OldBailey.xlsx", sheetName= "OldBailey", row.names= TRUE)
```

En este caso, y dentro del paréntesis de esta la función `write.xlsx` (<https://www.rdocumentation.org/packages/xlsx/versions/0.6.1/topics/write.xlsx>), estamos llamando a procesar la variable "OldBailey" con el argumento `x=` ; a la vez, indicamos que el archivo guardado debe llamarse "OldBailey" con la extensión ".xlsx" mediante el argumento `file=` ; además, damos el nombre "OldBailey" a la hoja de cálculo en que estarán los datos mediante `sheetName=` y, finalmente, establecemos que sí queremos (`TRUE` o verdadero) que los nombres de la fila en nuestra variable se guarden en el nuevo archivo. [N. de la T.]

Resumen y siguientes pasos

Este tutorial ha explorado lo básico de R para trabajar con datos tabulares de tu investigación. R puede ser una herramienta útil para la investigación en las Humanidades y las Ciencias Sociales porque el análisis de los datos es reproducible y te permite analizar datos de forma rápida sin tener que crear un sistema complicado. Ahora que conoces algunas funciones básicas de R puedes explorar algunas de las otras funciones del programa, incluyendo la computación estadística, la producción de gráficos y la creación de tus propias funciones.

Para más información sobre R, visita el [Manual de R](https://cran.r-project.org/doc/contrib/R-intro-1.1.0-espanol.1.pdf) (<https://cran.r-project.org/doc/contrib/R-intro-1.1.0-espanol.1.pdf>).

También hay numerosos tutoriales de R online, incluyendo:

- [R: A self-learn tutorial](http://web.archive.org/web/20191015004305/https://www.nceas.ucsb.edu/files/scicomp/Dloads/RProgramming/BestFirstRTutorial.pdf) (<http://web.archive.org/web/20191015004305/https://www.nceas.ucsb.edu/files/scicomp/Dloads/RProgramming/BestFirstRTutorial.pdf>) (en inglés) - este tutorial cubre varias funciones y provee ejercicios para practicar.
- [DataCamp Introducción a R](https://www.datacamp.com/community/open-courses/introduccion-a-r) (<https://www.datacamp.com/community/open-courses/introduccion-a-r>) (en español) - este es un curso online gratuito que te ofrece comentarios sobre tu código para ayudarte a identificar errores y aprender a escribir código más eficientemente.

Finalmente, un buen recurso para los historiadores digitales es el libro *Digital History Methods in R* (<http://dh-r.lincolnmullen.com>) de Lincoln Mullen.

Notas

1. Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1976), *Time Series Analysis, Forecasting and Control*. Third Edition. Holden-Day. Series G. ↵
2. Henderson and Velleman (1981), *Building multiple regression models interactively*. Biometrics, 37, 391-411. ↵

3. Nota de la traductora: un galón equivale a 3,78 litros y una milla a 1,6 kilómetros. ↵

ACERCA DEL AUTOR

Taryn Dewar tiene una maestría en Historia Pública (Western University) y es intérprete en el Centro de investigación de arenas de alquitrán en Fort McMurray, Alberta.

CITA SUGERIDA

Taryn Dewar, "Datos tabulares en R", traducido por Jennifer Isasi, *Programming Historian en español* 2 (2018), <https://doi.org/10.46430/phes0034>.

The Programming Historian en español (ISSN: 2517-5769) se publica con una licencia [CC-BY](https://creativecommons.org/licenses/by/4.0/deed.es) (<https://creativecommons.org/licenses/by/4.0/deed.es>).

Este proyecto es administrado por ProgHist Ltd, organización benéfica número [1195875](https://register-of-charities.charitycommission.gov.uk/charity-search/-/charity-details/5181272/charity-overview) (<https://register-of-charities.charitycommission.gov.uk/charity-search/-/charity-details/5181272/charity-overview>) y número de compañía [12192946](https://find-and-update.company-information.service.gov.uk/company/12192946) (<https://find-and-update.company-information.service.gov.uk/company/12192946>).


[ISSN 2397-2068 \(inglés\) \(/\)](#)


 [Alojado en GitHub \(https://github.com/programminghistorian/jekyll\)](https://github.com/programminghistorian/jekyll)


[ISSN 2517-5769 \(español\) \(/es\)](#)


[ISSN 2631-9462 \(francés\) \(/fr\)](#)

[ISSN 2753-9296 \(portugués\) \(/pt\)](#)


 Última actualización el 12 January 2023 (<https://github.com/programminghistorian/jekyll/commits/gh-pages>).

 Suscripción a RSS (<https://programminghistorian.org/feed.xml>).

 [Versiones anteriores \(https://github.com/programminghistorian/jekyll/commits/gh-pages/es/lecciones/datos-tabulares-en-r.md\)](https://github.com/programminghistorian/jekyll/commits/gh-pages/es/lecciones/datos-tabulares-en-r.md)

 [Envíanos tus comentarios \(/es/retroalimentacion\)](#)

[Política de retiro de lecciones \(/es/politica-retirada-lecciones\)](#)

 [Concordancia de traducción \(/translation-concordance\)](#)