Importar Datos

with readr, tibble, and tidyr

Guía Rápida

RStudio

El **tidyverse** de R está construido alrededor de los **datos ordenados** almacenados en **tibbles**, una versión mejorada del data frame.

El anverso de esta guía muestra como leer ficheros de texto en R con **readr**.

El reverso muestra como crear tibbles con **tibble** y como disponer datos ordenados con **tidyr**.

Otros tipos de datos

Prueba uno de los siguientes paquetes para importar otros tipos de ficheros

- **haven** - ficheros SPSS, Stata y SAS
- **readxl** - ficheros Excel (.xls y .xlsx)
- **DBI** - bases de datos
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Funciones Escritura

Salva **x**, un objeto R, en **path**, una ruta del fichero, con:

write_csv(x, path, na = "NA", append = FALSE, col_names = !append**)**

Tibble/df a fichero delimitado con coma.

write_delim(x, ruta, delim = " ", na = "NA", append = FALSE, col_names = !append**)**

Tibble/df to file with any delimiter.

write_excel_csv(x, path, na = "NA", append = FALSE, col_names = !append**)**

Tibble/df a CSV para Excel

write_file(x, path, append = FALSE**)**

Cadena a fichero.

write_lines(x, path, na = "NA", append = FALSE**)**

Vector cadena a fichero, un elemento por línea.

write_rds(x, path, compress = c("none", "gz", "bz2", "xz"), ...**)**

Objecto a fichero RDS.

write_tsv(x, path, na = "NA", append = FALSE, col_names = !append**)**

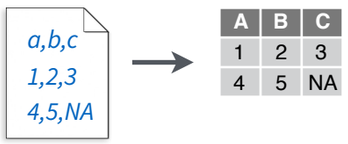
Tibble/df a ficheros delimitados por tab.

Funciones Lectura

Lectura de datos tabulares a tibbles

Estas funciones comparten los siguientes argumentos comunes:

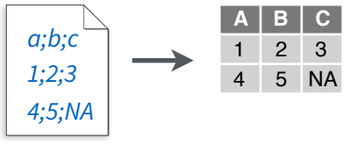
read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive()**)**



read_csv()

Lee ficheros delimitados por comas.

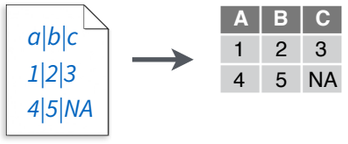
read_csv("file.csv")



read_csv2()

Lee ficheros delimitados por punto y coma.

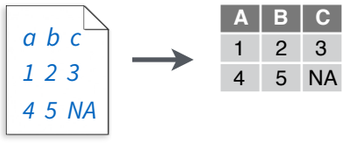
read_csv2("file2.csv")



read_delim(delim, quote = "\"", escape_backslash = FALSE, escape_double = TRUE**)**

Lee ficheros con cualquier delimitador.

read_delim("file.txt", delim = "|")



read_fwf(col_positions**)**

Lee ficheros con ancho fijo.

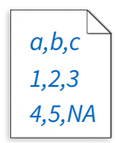
read_fwf("file.fwf", col_positions = c(1, 3, 5))

read_tsv()

Lee ficheros delimitados por tab. También **read_table()**.

read_tsv("file.tsv")

Argumentos útiles



Fichero ejemplo

write_csv (path = "file.csv", x = read_csv("a,b,c\n1,2,3\n4,5,NA"))

A	B	C
1	2	3
4	5	NA

Sin cabecera

*read_csv("file.csv", **col_names = FALSE**)*

x	y	z
A	B	C
1	2	3
4	5	NA

Proporciona cabecera

*read_csv("file.csv", **col_names = c("x", "y", "z")**)*

1	2	3
4	5	NA

Salta líneas

*read_csv("file.csv", **skip = 1**)*

A	B	C
1	2	3

Lee un subconjunto

*read_csv("file.csv", **n_max = 1**)*

A	B	C
1	2	3
NA	NA	NA

Valores Faltantes

*read_csv("file.csv", **na = c("4", "5", ".")**)*

Lectura de datos no tabulares

read_file(file, locale = default_locale()**)**

Lee un fichero en una sola cadena.

read_file_raw(file**)**

Lee un fichero en un vector raw.

read_lines(file, skip = 0, n_max = -1L, locale = default_locale(), na = character(), progress = interactive()**)**

Lee cada línea en una cadena.

read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive()**)**

Lee cada línea en un vector raw.

read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive()**)**

Ficheros de log estilo Apache.

Filtrando tipos de datos

Las funciones de readr interpretan los tipos de cada columna y convierten los tipos de forma apropiada (pero NO convertirán cadenas a factores automáticamente).

Un mensaje muestra el tipo de columna en el resultado.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

age es un entero

sex es un caracter

earn es un doble (numérico)

1. Usa **problems()** para diagnosticar problemas

x <- read_csv("file.csv"); problems(x)

2. Usa col_ function para guiar el filtrado

- **col_guess()** - por defecto
- **col_character()**
- **col_double()**
- **col_euro_double()**
- **col_datetime(**format = ""**)** También **col_date(**format = ""**)** and **col_time(**format = ""**)**
- **col_factor(**levels, ordered = FALSE**)**
- **col_integer()**
- **col_logical()**
- **col_number()**
- **col_numeric()**
- **col_skip()**

```
x <- read_csv("file.csv", col_types = cols(
  A = col_double(),
  B = col_logical(),
  C = col_factor()
))
```

3. Para el resto, los lee como vectores carácter y luego los filtra con la función parse_ .

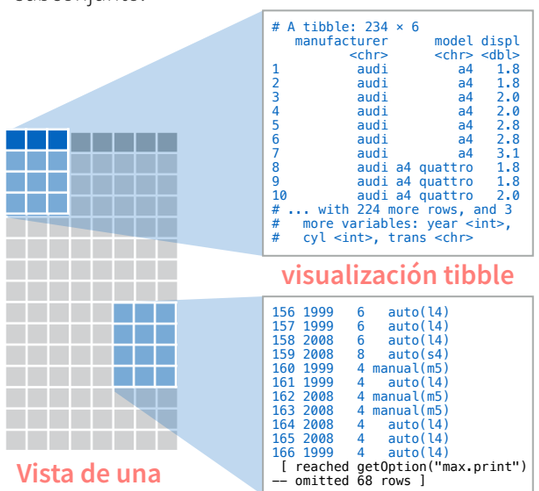
- **parse_guess(**x, na = c("", "NA"), locale = default_locale()**)**
- **parse_character(**x, na = c("", "NA"), locale = default_locale()**)**
- **parse_datetime(**x, format = "", na = c("", "NA"), locale = default_locale()**)** Also **parse_date()** and **parse_time()**
- **parse_double(**x, na = c("", "NA"), locale = default_locale()**)**
- **parse_factor(**x, levels, ordered = FALSE, na = c("", "NA"), locale = default_locale()**)**
- **parse_integer(**x, na = c("", "NA"), locale = default_locale()**)**
- **parse_logical(**x, na = c("", "NA"), locale = default_locale()**)**
- **parse_number(**x, na = c("", "NA"), locale = default_locale()**)**

x$A <- parse_number(x$A)

Tibbles - un data frame mejorado

El paquete **tibble** proporciona una nueva clase S3 para almacenar datos tabulares, el tibble. Tibbles heredan la clase del data frame, pero mejora dos comportamientos:

- **Visualiza** - Cuando se imprime un tibble, R proporciona una vista concisa de los datos que se ajustan en una pantalla.
- **Subconjunto** - [siempre devuelve un nuevo tibie , [[y $ siempre devuelve un vector.
- **No emparejado parcial** - Se debe usar el nombre completo de las columnas cuando se selecciona un subconjunto.



```
# A tibble: 234 × 6
   manufacturer      model displ
          <chr>      <chr> <dbl>
1          audi         a4   1.8
2          audi         a4   1.8
3          audi         a4   2.0
4          audi         a4   2.0
5          audi         a4   2.8
6          audi         a4   2.8
7          audi         a4   3.1
8          audi a4 quattro   1.8
9          audi a4 quattro   1.8
10         audi a4 quattro   2.0
# ... with 224 more rows, and 3
#   more variables: year <int>,
#   cyl <int>, trans <chr>
```

visualización tibble

Vista de una tabla larga

```
156 1999   6    auto(l4)
157 1999   6    auto(l4)
158 2008   6    auto(l4)
159 2008   8    auto(s4)
160 1999   4 manual(m5)
161 1999   4    auto(l4)
162 2008   4 manual(m5)
163 2008   4 manual(m5)
164 2008   4    auto(l4)
165 2008   4    auto(l4)
166 1999   4    auto(l4)
 [ reached getOption("max.print")
-- omitted 68 rows ]
```

visualización data frame

- La apariencia por defecto se controla con las opciones:

  options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)**

- La vista del conjunto completo de datos con **View(**x, title**)** o **glimpse(**x, width = NULL, …**)**
- Revertir a data frame con **as.data.frame()** (requerido por algunos paquetes antiguos)

Construir un tibble en dos formas

tibble(…**)**
Construye por columnas.
tibble(x = 1:3, y = c("a", "b", "c"))

tribble(…**)**
Construye por filas.
tribble(
~x, ~y,
1, "a",
2, "b",
3, "c")

Ambos crean este tibble

```
A tibble: 3 × 2
      x     y
  <int> <dbl>
1     1     a
2     2     b
3     3     c
```

as_tibble(x, …**)** Convierte data frame a tibble.

enframe(x, name = "name", value = "value"**)**
Convierte un vector con nombre a un tibble con una columna con nombre y una columna valor.

is_tibble(x**)** Compueba si x es un tibble.

Datos Ordenados con tidyr

Datos Tidy es una forma de organizar datos tabulares. Proporciona una estructura consistente de datos entre paquetes.

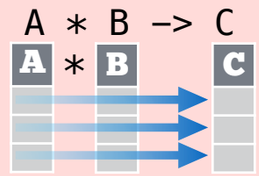
Una tabla es tidy si:

Cada **variable** está en su propia **columna**

Cada **observación** o **caso**, está en su propia **fila**

Datos Tidy:

Permite el acceso a las variables como vectores



Preserva los casos en las operaciones vectorizadas

Remodelado de Datos - cambia la disposición de los valores en una tabla

Usa **gather()** y **spread()** para reorganizar los valores de una tabla en una nueva disposición. Usan la idea de una columna clave: par valor columna.

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE**)**

Gather mueve los nombres de las columnas a una columna llave, reuniendo los valores de las columnas en una única columna.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

llave valor

gather(table4a, `1999`, `2000`, key = "year", value = "cases")

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL**)**

Spread mueve el valor único de una columna llave a nombres de columna, repartiendo los valores de una columna entre las nuevas columnas que resultan.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

llave valor

spread(table2, type, count)

Gestión de Datos Faltantes

drop_na(data, ...**)**
Elimina columnas con NAs.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(x, x2)

fill(data, ..., .direction = c("down", "up"**)**
Completa los NA's en … columnas con los valores no-NA más cercanos.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(x, x2)

replace_na(data, replace = list(), ...**)**
Reemplaza NA's por columna.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(x,list(x2 = 2), x2)

Expansión de Tablas - crea tablas rápidamente con combinaciones de valores

complete(data, ..., fill = list()**)**
Añade a los datos combinaciones faltantes de los valores listados en …
complete(mtcars, cyl, gear, carb)

expand(data, ...**)**
Crea un nuevo tibble con todas las posibles combinaciones de valores de las variables listadas en …
expand(mtcars, cyl, gear, carb)

Separar y Combinar Celdas

Usa estas funciones para separar o combinar celdas en valores individuales, aislados.

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...**)**

Separa cada celda de una columna para crear varias columnas

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

separate_rows(table3, rate, into = c("cases", "pop"))

separate_rows(data, ..., sep = "[^[:alnum:].]+", convert = FALSE**)**

Separa cada celda en una columna para crear varias filas. También **separate_rows_()**.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

separate_rows(table3, rate)

unite(data, col, ..., sep = "_", remove = TRUE**)**

Collapsa celdas de varias columnas para crear una única columna.

table5

country	century	year
Afghan	19	99
Afghan	20	0
Brazil	19	99
Brazil	20	0
China	19	99
China	20	0

→

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

unite(table5, century, year, col = "year", sep = "")