

Manejo de datos con dplyr

Ronny A. Hernández Mora.

 @RonnyHdezM

 ronnyhdez

 <http://ronnyhdez.rbind.io/>

Material del curso

README.md

¡Bienvenido al curso de ciencia de datos con R!

Este es un curso libre y gratuito que puede usar para dar sus primeros pasos con el lenguaje de programación R.



Artwork by @allison_horst

Material del curso

Sesión	Presentación	Video
0-Preparación	No hay	https://www.youtube.com/watch?v=NCvJwJSMq60
1- Introducción a las herramientas	Por subir	Por subir



https://github.com/ronnyhdez/curso_ciencia_datos_r

main 4 branches 0 tags

Go to file Add file **Code**

ronnyhdez Merge pull request #11 from ronnyhdez/T8

- img Ref #1 estructura del curso
- presentaciones Ref #8 material presentacion
- sesion_01 Ref 1 material sesion 1
- sesion_02 Ref #8 orden en script segunda ses
- .gitignore Ref #2 materiales sesion 2

Clone

HTTPS SSH GitHub CLI

https://github.com/ronnyhdez/curso_cie

Use Git or checkout with SVN using the web URL.

Download ZIP

10 days ago

¿Qué queremos de la sesión de hoy?

- Cómo importar datos
- Comprender el concepto de datos ordenados
- Uso de verbos de dplyr para manipular datos





Importar datos

Hoja de referencia

Data Import :: CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.



The front side of this sheet shows how to read text files into R with **readr**.



The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidyr**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save **x**, an R object, to **path**, a file path, as:

Comma delimited file
write_csv(**x**, **path**, **na** = "NA", **append** = FALSE, **col_names** = **lappend**)

File with arbitrary delimiter
write_delim(**x**, **path**, **delim** = " ", **na** = "NA", **append** = FALSE, **col_names** = **lappend**)

CSV for excel
write_excel_csv(**x**, **path**, **na** = "NA", **append** = FALSE, **col_names** = **lappend**)

String to file
write_file(**x**, **path**, **append** = FALSE)

String vector to file, one element per line
write_lines(**x**, **path**, **na** = "NA", **append** = FALSE)

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

Tab delimited files
write_tsv(**x**, **path**, **na** = "NA", **append** = FALSE, **col_names** = **lappend**)

Read Tabular Data - These functions share the common arguments:

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**())

Comma Delimited Files
read_csv("file.csv")
To make file.csv run:
write_file(**x** = "a,b,c(1,2,3)n4,5,NA", **path** = "file.csv")

Semi-colon Delimited Files
read_csv2("file2.csv")
write_file(**x** = "a;b;c(1;2;3)n4;5;NA", **path** = "file2.csv")

Files with Any Delimiter
read_delim("file.txt", **delim** = "|")
write_file(**x** = "a|b|c(1|2|3)n4|5|NA", **path** = "file.txt")

Fixed Width Files
read_fwf("file.fwf", **col_positions** = c(1, 3, 5))
write_file(**x** = "a b c(1 2 3)n4 5 NA", **path** = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv") Also **read_table**()
write_file(**x** = "a\tb\tc(1\t2\t3)n4\t5\tNA", **path** = "file.tsv")

USEFUL ARGUMENTS

Example file
write_file("a,b,c(1,2,3)n4,5,NA","file.csv")
f <- "file.csv"

No header
read_csv(**f**, **col_names** = FALSE)

Provide header
read_csv(**f**, **col_names** = c("x", "y", "z"))

Skip lines
read_csv(**f**, **skip** = 1)

Read in a subset
read_csv(**f**, **n_max** = 1)

Missing Values
read_csv(**f**, **na** = c("1", "m"))

Read Non-Tabular Data

Read a file into a single string
read_file(**file**, **locale** = **default_locale**())

Read each line into its own string
read_lines(**file**, **skip** = 0, **n_max** = 1L, **na** = **character**(), **locale** = **default_locale**(), **progress** = **interactive**())

Read a file into a raw vector
read_file_raw(**file**)

Read each line into a raw vector
read_lines_raw(**file**, **skip** = 0, **n_max** = 1L, **progress** = **interactive**())

Read Apache style log files
read_log(**file**, **col_names** = FALSE, **col_types** = NULL, **skip** = 0, **n_max** = -1, **progress** = **interactive**())



Data types

readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

A message shows the type of each column in the result.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
#> # A tibble: 1 x 3
#>   age sex   earn
#>   <int> <chr> <dbl>
#> 1   23 m    45678
```

1. Use **problems()** to diagnose problems.

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

2. Use a **col_*** function to guide parsing.

- **col_guess()** the default
 - **col_character()**
 - **col_double()**, **col_euro_double()**
 - **col_datetime**(**format** = "%Y-%m-%d") Also **col_date**(**format** = "%Y-%m-%d")
 - **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. Else, read in as character vectors then parse with a **parse_*** function.

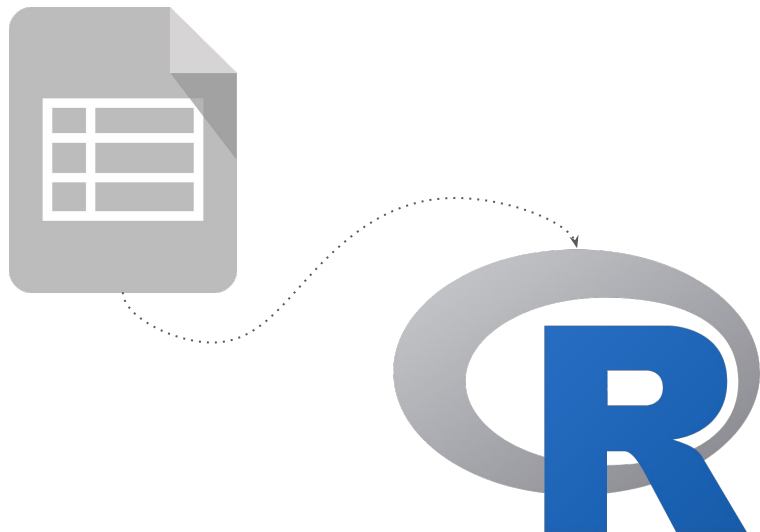
- **parse_guess()**
 - **parse_character()**
 - **parse_datetime()** Also **parse_date()** and **parse_time()**
 - **parse_double()**
 - **parse_factor()**
 - **parse_integer()**
 - **parse_logical()**
 - **parse_number()**
- xSA** <- **parse_number**(**xSA**)



RStudio is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at [tidyverse.org](https://www.rstudio.com) • readr 1.1.0 • tibble 1.2.12 • tidyr 0.6.0 • Updated: 2021-03

¿Cómo llevar datos de un archivo a mi sesión de R?

```
read_csv( "direccion_al_archivo" )
```



```
read_*( )
```




<https://readr.tidyverse.org/>



<https://readxl.tidyverse.org/>



Café 20:00 a 20:10

dplyr : go wrangling



Domando datos con dplyr

Hoja de referencia

Data Transformation with dplyr : : CHEAT SHEET



dplyr functions work with pipes and expect tidy data. In tidy data:



Each **variable** is in its own **column**



Each **observation**, or **case**, is in its own **row**



$x \%>\% f(y)$ becomes $f(x, y)$

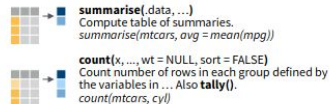


pipes

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function



Group Cases

Use **group_by**(data, ..., add = FALSE) to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.



Use **rowwise**(data, ...) to group data into individual rows. dplyr functions will compute results for each row. Also used to apply functions to list-columns without purrr functions.

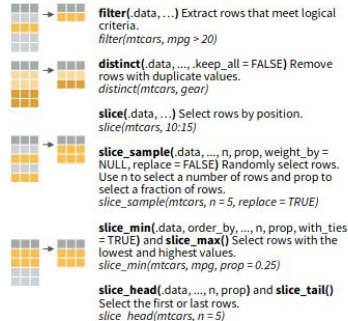


ungroup(x, ...) Returns ungrouped copy of table.
`ungroup(g_mtcars)`

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

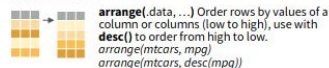


Logical and boolean operators to use with filter()

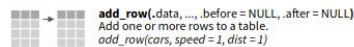
< <= is.na() %in% | xor()
> >= is.na() ! &

See ?base::Logic and ?Comparison for help.

ARRANGE CASES



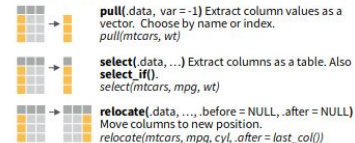
ADD CASES



Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

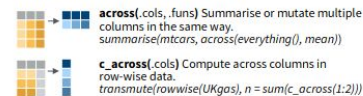


Use these helpers with select() and across()

e.g. `select(mtcars, mpg:cyl)`

contains(match) **num_range**(prefix, range) **starts_with**(match) **ends_with**(match) **one_of**(...) **matches**(match) **everything**()

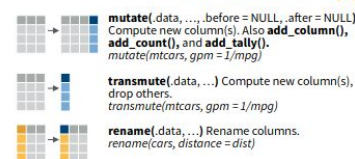
MANIPULATE MULTIPLE VARIABLES AT ONCE



MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

vectorized function



RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = c("dplyr", "tidbly")) • dplyr 1.0.6 • tidbly 3.1.2 • Updated: 2021-06

`select()`

`filter()`

`mutate()`

`group_by()`

`summarise()`

primera_instruccion %>%

segunda_instruccion %>%

tercera_instruccion



Proyecto

[UNA TRANSPARENTE](#) [INICIO](#) [COMISIÓN](#) [BUSCAR](#) [CONTACTO](#) [MAPA DEL SITIO](#)



Evaluación de desempeño

 Ámbito talento humano  15 Mayo 2017  Última actualización: 19 Febrero 2021  Visto: 8151

[Twitter](#) [Facebook](#) [LinkedIn](#) [Google+](#)

La Universidad Nacional realiza evaluaciones de desempeño de su personal tanto para los académicos como los funcionarios y funcionarias que laboran en el ámbito administrativo.

A continuación, se brinda la información de los resultados de dichas evaluaciones:

Evaluación del Desempeño Docente en la Universidad Nacional

La evaluación del desempeño docente en la Universidad Nacional constituye una práctica regular y se visualiza como una vía para el mejoramiento continuo; misma que permite tener congruencia con las exigencias de la realidad y orientar la toma de decisiones informadas hacia la búsqueda constante de la calidad. A continuación, se presenta los informes de las evaluaciones de desempeño según facultad y nombre del académico, centro y sede.

Resultados de la evaluación del desempeño docente, percepción por parte del estudiantado.

[DESCARGAR MASIVA >](#) [TUTORIAL PARA REALIZAR LA EVALUACIÓN DOCENTE >](#)

Exploración de
datos abiertos
de la UNA sobre
evaluación del
desempeño

https://www.transparencia.una.ac.cr/index.php?option=com_content&view=article&id=333&Itemid=787

Programación en C++
Algoritmos, estructuras de datos y objetos

L. Joyanes Aguilar

EI385

Learning Python

FIFTH EDITION

Lutz

LINUX BASICS FOR HACKERS

O'REILLY

Fundamentals of Data Visualization

Wilke

O'REILLY

Mastering Ubuntu Server
Third Edition

Jay LaCroix

<|

R for Data Science

John Fox
John Fox

O'REILLY

THINK LIKE A PROGRAMMER

SPRAT



The Pragmatic
Programmer

20th ANNIVERSARY
EDITION



Zuur • Ieno • Walker
Saveliev • Smith



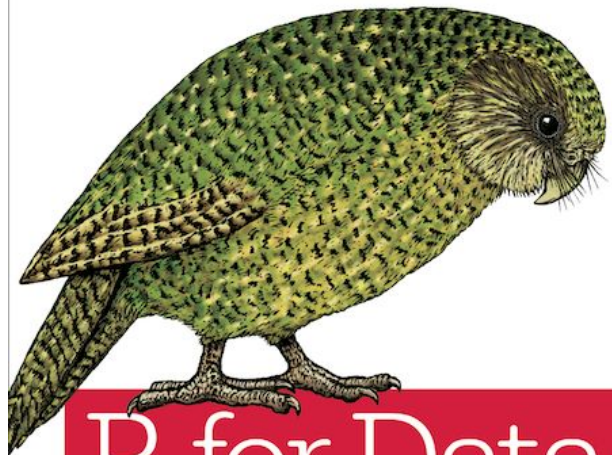
Mixed Effects Models
Extensions in Ecology with R

Wichham

Advanced R

Recursos

O'REILLY®



R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA


Hadley Wickham &
Garrett Grolemund

<https://es.r4ds.hadley.nz/>



¡Gracias !

Ronny A. Hernández Mora.

 @RonnyHdezM

 ronnyhdez

 <http://ronnyhdez.rbind.io/>

ronny.hernandezm@gmail.com