




Datos ordenados y fechas

Ronny A. Hernández Mora.

 @RonnyHdezM

 ronnyhdez

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



¡Grabar!

Sesión	Temas	Horario
1. Introducción a R	<ul style="list-style-type: none"> - ¿Qué es R? - ¿Qué es RStudio? - Sintaxis R 	Miércoles de 19:00 a 21:00 26 de mayo 2021
2. Estructuras de datos en R	<ul style="list-style-type: none"> - Clases - Estructuras de datos - Coerción 	Miércoles de 19:00 a 21:00 2 de junio 2021
3. Introducción a Rmarkdown	<ul style="list-style-type: none"> - ¿Qué es markdown? - Estructura de archivos .Rmd - Informes - 	Miércoles de 19:00 a 21:00 9 de junio 2021
4. Manejo de datos con dplyr	<ul style="list-style-type: none"> - Importar datos - Funciones básicas de dplyr 	Miércoles de 19:00 a 21:00 16 de junio 2021
5. Manejo de datos con tidyr y fechas con lubridate	<ul style="list-style-type: none"> - Concepto de tidydata - Funciones en tidyr - Fechas en R con lubridate 	Miércoles de 19:00 a 21:00 23 de junio 2021
6. Visualización de datos con ggplot2	<ul style="list-style-type: none"> - Gramática de gráficos - Funciones de ggplot2 	Miércoles de 19:00 a 21:00 30 de junio 2021
7. Presentaciones de proyectos	<ul style="list-style-type: none"> - Estudiantes presentan un análisis de datos en un informe construido con Rmarkdown. 	Miércoles de 19:00 a 21:00 7 de julio 2021

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?

- Entender qué son datos ordenados
- Saber usar funciones de tidyr para ordenar datos
- Comprender el uso de fechas





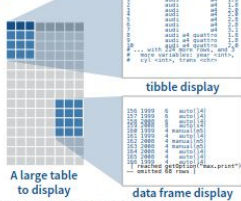
Datos ordenados

Hoja de referencia

Tibbles - an enhanced data frame

The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve three behaviors:

- Subsetting** - Always returns a new tibble, `[[` and `$` always return a vector.
- No partial matching** - You must use full column names when subsetting.
- Display** - When you print a tibble, R provides a concise view of the data that fits on one screen.



- Control the default appearance with options:
`options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)`
- View full data set with `View()` or `glimpse()`
- Revert to data frame with `as.data.frame()`

CONSTRUCT A TIBBLE IN TWO WAYS

`tibble(...)`
Construct by columns.
`tibble(x = 1:3, y = c("a", "b", "c"))`

`tribble(...)`
Construct by rows.
`tribble(~x, ~y, ~c("a", "b", "c"))`

Both make this tibble

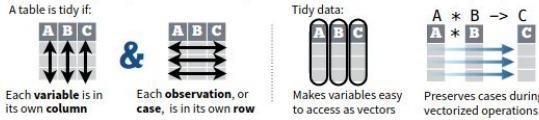
x	y
1	"a"
2	"b"
3	"c"

`as_tibble(x, ...)` Convert data frame to tibble.
`enframe(x, name = "name", value = "value")` Convert named vector to a tibble
`is_tibble(x)` Test whether x is a tibble.



Tidy Data with tidy

Tidy data is a way to organize tabular data. It provides a consistent data structure across packages.

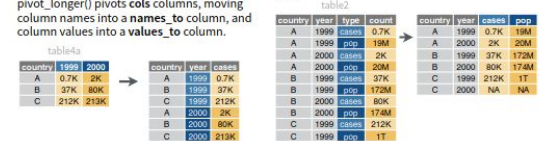


Reshape Data - change the layout of values in a table

Use `pivot_longer()` and `pivot_wider()` to reorganize the values of a table into a new layout.

pivot_longer() (data, cols, names, to = "name", names_prefix = NULL, names_sep = NULL, names_pattern = NULL, names_ptypes = list(), names_transform = list(), names_repair = "check_unique", values_to = "value", values_drop_na = FALSE, values_ptypes = list(), values_transform = list(), ...)

pivot_wider() (data, id_cols = NULL, names, from = "name", names_prefix = "", names_sep = "", names_glue = NULL, names_sort = FALSE, names_repair = "check_unique", values_from = value, values_fill = NULL, values_fn = NULL, ...)



`pivot_longer(table4a, cols = 2:3, names_to = "year", values_to = "cases")`
`pivot_wider(table2, names_from = type, values_from = count)`

Handle Missing Values

drop_na(data, ...)
Drop rows containing NA's in ... columns.

fill(data, ..., direction = c("down", "up"))
Fill in NA's in ... columns with most recent non-NA values.

replace_na(data, replace = list(), ...)
Replace NA's by column.

Expand Tables - quickly create tables with combinations of values

complete(data, ..., fill = list())
Adds to the data missing combinations of the values of the variables listed in ...
`complete(mtcars, cyl, gear, carb)`

expand(data, ...)
Create new tibble with all possible combinations of the values of the variables listed in ...
`expand(mtcars, cyl, gear, carb)`

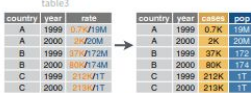
Split Cells

Use these functions to split or combine cells into individual, isolated values.



separate(data, col, into, sep = "[a-zA-Zm:;]", *, remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...)

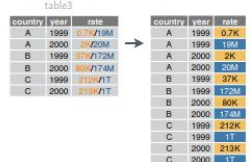
Separate each cell in a column to make several columns.



`separate(table3, rate, sep = "/", into = c("cases", "pop"))`

separate_rows(data, ..., sep = "[a-zA-Zm:;]", *, convert = FALSE)

Separate each cell in a column to make several rows.



`separate_rows(table3, rate, sep = "/")`

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

Collapse cells across several columns to make a single column.



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

¿Qué son datos ordenados?



curso	fecha	calificacion	estudiantes
matematica	2020-05-28	excelente	34
historia del arte	2020-06-04	regular	20
computacion	2020-06-12	bueno	28

cada **fila** es una **observación**

cada **columna** es una **variable**

- Cada **variable** debe de tener su propia **columna**.
- Cada **observación** debe de tener su propia **fila**.
- Cada **valor** debe de tener su propia **celda**.

¿Estos son datos ordenados?

curso	fecha	calificacion	estudiantes/respuestas
matematica	2020-05-28	excelente	34/20
historia del arte	2020-06-04	regular	20/18
computacion	2020-06-12	bueno	28/12

¿Estos son datos ordenados?

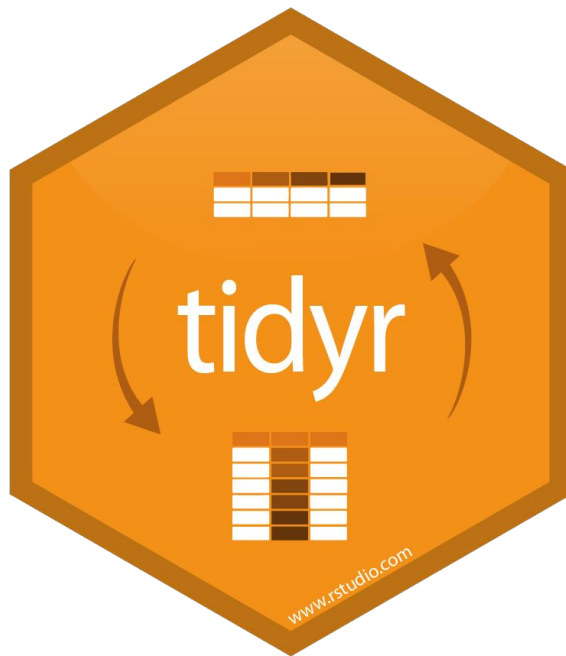
curso	fecha	calificacion	tipo	total
matematica	2020-05-28	excelente	estudiantes	34
matematica	2020-05-28	excelente	respuestas	20
historia del arte	2020-06-04	regular	estudiantes	20
historia del arte	2020-06-04	regular	respuestas	18
computacion	2020-06-12	bueno	estudiantes	28
computacion	2020-06-12	bueno	respuestas	12

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

```
pivot_longer( argumentos )
```

```
pivot_wider( argumentos )
```





<https://tidyr.tidyverse.org>



<https://www.garrickadenbuie.com/project/tidyexplain/>



Café 20:00 a 20:10



Fechas con lubridate

Hoja de referencia

Dates and times with lubridate : : CHEAT SHEET



Date-times



2017-11-28 12:00:00
A **date-time** is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC
`dt <- as_datetime(1511870400)`
"2017-11-28 12:00:00 UTC"

2017-11-28
A **date** is a day stored as the number of days since 1970-01-01
`d <- as_date(17498)`
"2017-11-28"

12:00:00
An **hms** is a **time** stored as the number of seconds since 00:00:00
`t <- hms-as.hms(85)`
"00:01:25"

PARSE DATE-TIMES (Convert strings or numbers to date-times)

1. Identify the order of the year (**y**), month (**m**), day (**d**), hour (**h**), minute (**m**) and second (**s**) elements in your data.
2. Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00

`ymd_hms()` `ymd_hm()` `ymd_h()`
`ymd_hms("2017-11-28T14:02:00")`

2017-22-12 10:00:00

`ydm_hms()` `ydm_hm()` `ydm_h()`
`ydm_hms("2017-22-12 10:00:00")`

11/28/2017 1:02:03

`mdy_hms()` `mdy_hm()` `mdy_h()`
`mdy_hms("11/28/2017 1:02:03")`

1 Jan 2017 23:59:59

`dmy_hms()` `dmy_hm()` `dmy_h()`
`dmy_hms("1 Jan 2017 23:59:59")`

20170131

`ymd()` `ydm()` `ymd(20170131)`

July 4th, 2000

`mdy()` `myd()` `mdy("July 4th, 2000")`

4th of July 99

`dmy()` `dym()` `dmy("4th of July '99")`

2001: Q3

`yq()` `Q` for quarter. `yq("2001: Q3")`

2 01

`hms-as.hms()` Also `lubridate-hms()`, `hm()` and `ms()`, which return periods: `hms`: `hms(sec = 0, min = 1, hours = 2)`

2017.5

`date_decimal(decimal, tz = "UTC")`
`date_decimal(2017.5)`



`now(tzone = "")` Current time in `tz` (defaults to system `tz`). `now()`

`today(tzone = "")` Current date in a `tz` (defaults to system `tz`). `today()`

`fast_strptime()` Faster `strptime`.
`fast_strptime("9/1/01", "%y/%m/%d")`

`parse_date_time()` Easier `strptime`.
`parse_date_time("9/1/01", "ymd")`

GET AND SET COMPONENTS

Use an accessor function to get a component. Assign into an accessor function to change a component in place.

`d ## "2017-11-28"`
`day(d) ## 28`
`day(d) <- 1`
`d ## "2017-11-01"`

2018-01-31 11:59:59

`date(x)` Date component. `date(dt)`

2018-01-31 11:59:59

`year(x)` Year. `year(dt)`

2018-01-31 11:59:59

`isoyear(x)` The ISO 8601 year.

2018-01-31 11:59:59

`epiyear(x)` Epidemiological year.

2018-01-31 11:59:59

`month(x, label, abbr)` Month. `month(dt)`

2018-01-31 11:59:59

`day(x)` Day of month. `day(dt)`

2018-01-31 11:59:59

`wday(x, label, abbr)` Day of week.

2018-01-31 11:59:59

`qday(x)` Day of quarter.

2018-01-31 11:59:59

`hour(x)` Hour. `hour(dt)`

2018-01-31 11:59:59

`minute(x)` Minutes. `minute(dt)`

2018-01-31 11:59:59

`second(x)` Seconds. `second(dt)`

2018-01-31 11:59:59

`week(x)` Week of the year. `week(dt)`

2018-01-31 11:59:59

`isoweek(x)` ISO 8601 week.

2018-01-31 11:59:59

`epiweek()` Epidemiological week.

2018-01-31 11:59:59

`quarter(x, with_year = FALSE)`

2018-01-31 11:59:59

`quarter(x, quarter(dt))`

2018-01-31 11:59:59

`semester(x, with_year = FALSE)`

2018-01-31 11:59:59

`semester(x, semester(dt))`

2018-01-31 11:59:59

`am(x)` Is it in the am? `am(dt)`

2018-01-31 11:59:59

`pm(x)` Is it in the pm? `pm(dt)`

2018-01-31 11:59:59

`dst(x)` Is it daylight savings? `dst(dt)`

2018-01-31 11:59:59

`leap_year(x)` Is it a leap year?

2018-01-31 11:59:59

`leap_year(dt)`

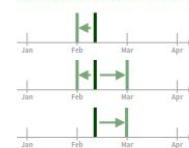
2018-01-31 11:59:59

`update(object, ..., simple = FALSE)`

2018-01-31 11:59:59

`update(dt, mday = 2, hour = 1)`

Round Date-times



floor_date(x, unit = "second")
Round down to nearest unit.
`floor_date(dt, unit = "month")`

round_date(x, unit = "second")
Round to nearest unit.
`round_date(dt, unit = "month")`

ceiling_date(x, unit = "second")
Round up to nearest unit.
`ceiling_date(dt, unit = "month")`

rollback(dates, roll, to, first = FALSE, preserve_hms = TRUE)
Roll back to last day of previous month. `rollback(dt)`

Stamp Date-times

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also `stamp_date()` and `stamp_time()`.

1. Derive a template, create a function
`sf <- stamp("Created Sunday, Jan 17, 1999 3:34")`
2. Apply the template to dates
`sfymd("2010-04-05")`
[1] "Created Monday, Apr 05, 2010 00:00"

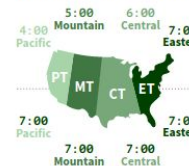
Tip: use a date with day = 12

Time Zones

R recognizes ~600 time zones. Each encodes the time zone, Daylight Savings Time, and historical calendar variations for an area. R assigns one time zone per vector.

Use the **UTC** time zone to avoid Daylight Savings.

OlsonNames() Returns a list of valid time zone names. `OlsonNames()`



with_tz(time, tzone = "") Get the same date-time in a new time zone (a new clock time).
`with_tz(dt, "US/Pacific")`

force_tz(time, tzone = "") Get the same clock time in a new time zone (a new date-time).
`force_tz(dt, "US/Pacific")`



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

SPRINGER



The
Pragmatic
Programmer

20th ANNIVERSARY
EDITION



Zuur • Ieno • Walker
Saveliev • Smith



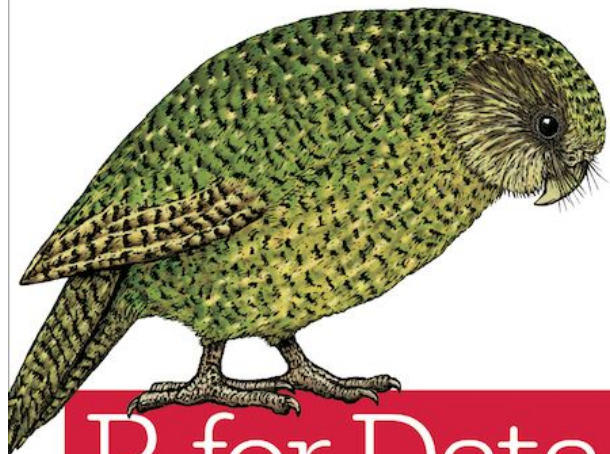
Mixed Effects Models
Extensions in Ecology with R

Wichham

Advanced R

Recursos

O'REILLY®



R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA


Hadley Wickham &
Garrett Golemund

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



¡Gracias !

Ronny A. Hernández Mora.

 @RonnyHdezM

 ronnyhdez

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

ronny.hernandezm@gmail.com