```
library(dplyr)
library(magrittr)

Curso_R %>% filter(city == 'Corrientes')
```

BIENVENIDOS AL TALLER

APRENDER R DESDE CERO

COORDINADORA UNNE:
MG. SONIA MARIÑO

DICTANTES

- **PHD. ROXANA VILLAFAÑE**
- **LIC. EN SISTEMAS PATRICIA A. LOTO**

Lunes 26 de agosto de 2019.

PARTE 1
REPASO

IMPORTACIÓN DE DATOS

Breve Repaso

¿Cómo instalamos un paquete?

- **install.packages (“*nombre_del_paquete*")**
- **library (*nombre_del_paquete*)**

Importación de datos en R

#Por medio de read.csv()

- Iris <- **read.csv("E:/DATASETS/iris.csv")**
- View(iris)

#Por medio de read.table()

- mtcars <- **read.table**("E:/DATASETS/mtcars.txt")
- mtcars <-**read.table**("E:/DATASETS/mtcars.txt", header=TRUE)
- View(mtcars)

Importación de datos en R

#También podemos importar otro tipo de datos mediante el paquete readxl. Permite leer archivos con formato (.xls and .xlsx) into R

Ejemplo:

- **install.packages ("readxl")**
- **library (readxl)**
- **estadis <- read_xl ("estadistica2009.xlsx")**

Hoy hablamos sobre…

PARTE 1: DATOS ORDENADOS CON TIDYR

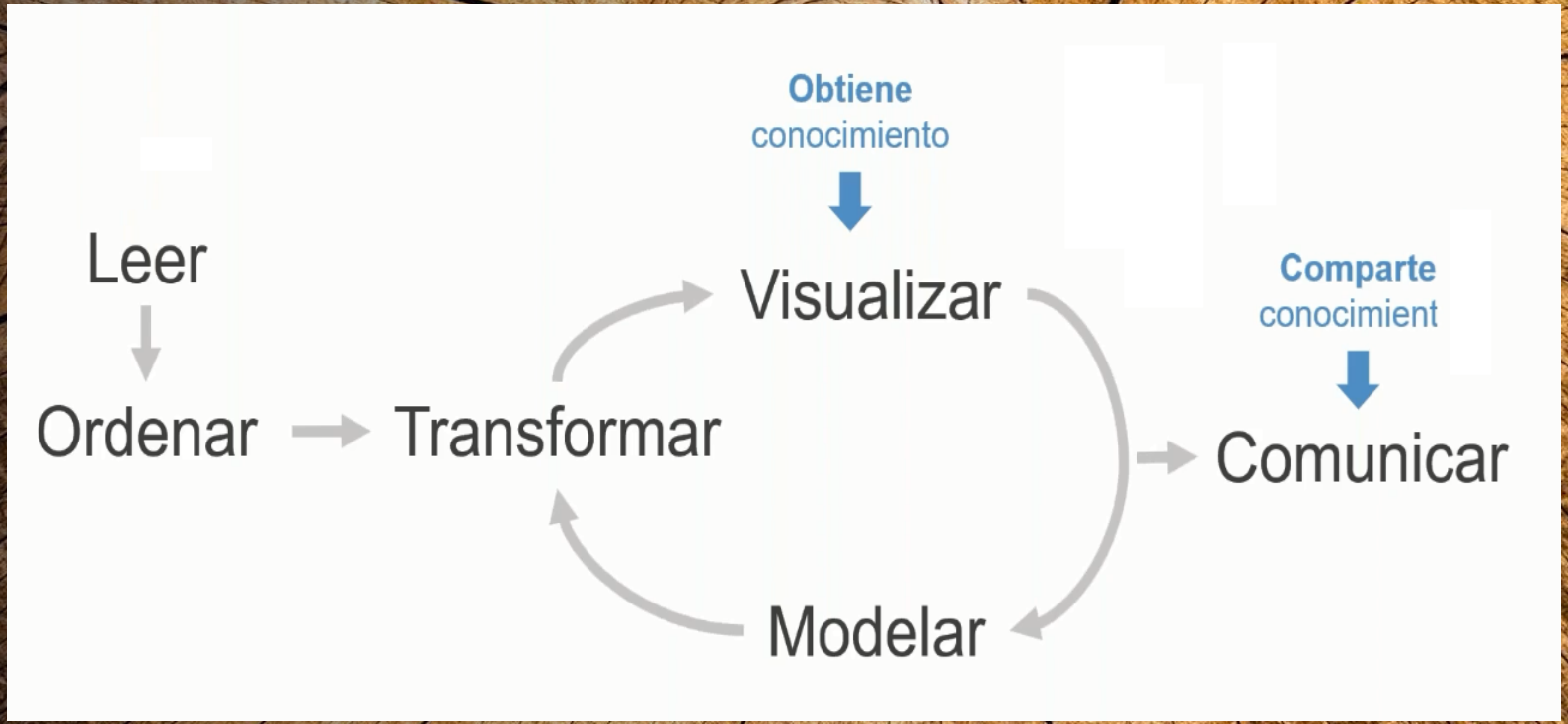
Funciones más importantes

PARTE 2: MANIPULACIÓN DE DATOS CON DPLYR

Funciones más importantes

Operador %>%

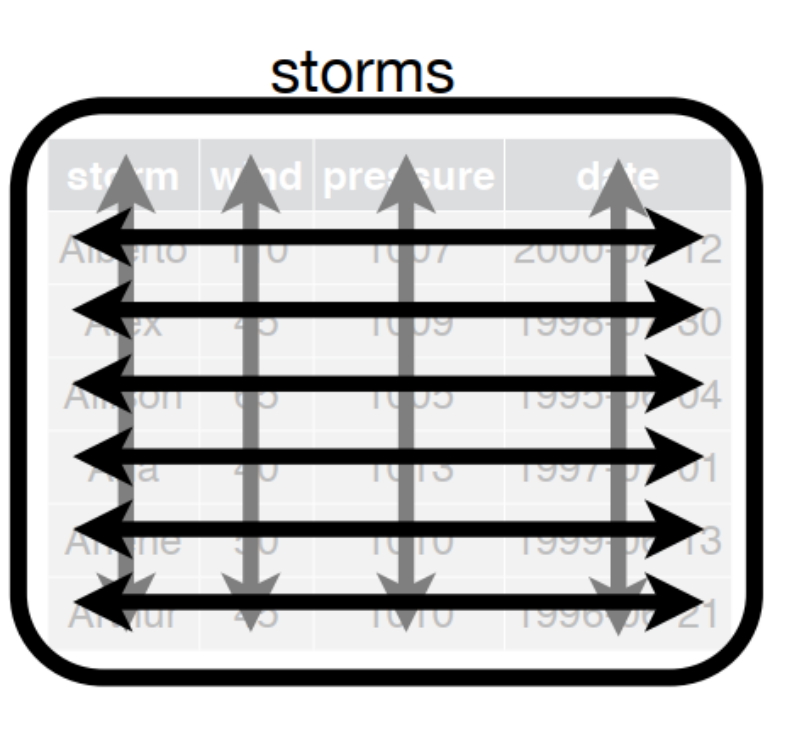
DATA SCIENCE WORKFLOW



PARTE 2

ORDENAR DATOS CON tidyr

TIDY DATA



1. Cada **variable** está en su **columna**
2. Cada **observación** está en una **fila**
3. Cada tipo de observación está en una tabla aparte

LAS BASES DE DATOS PARA NUESTRO TRABAJO

install.packages ("devtools")

devtools::install_github("rstudio/EDAWR")

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

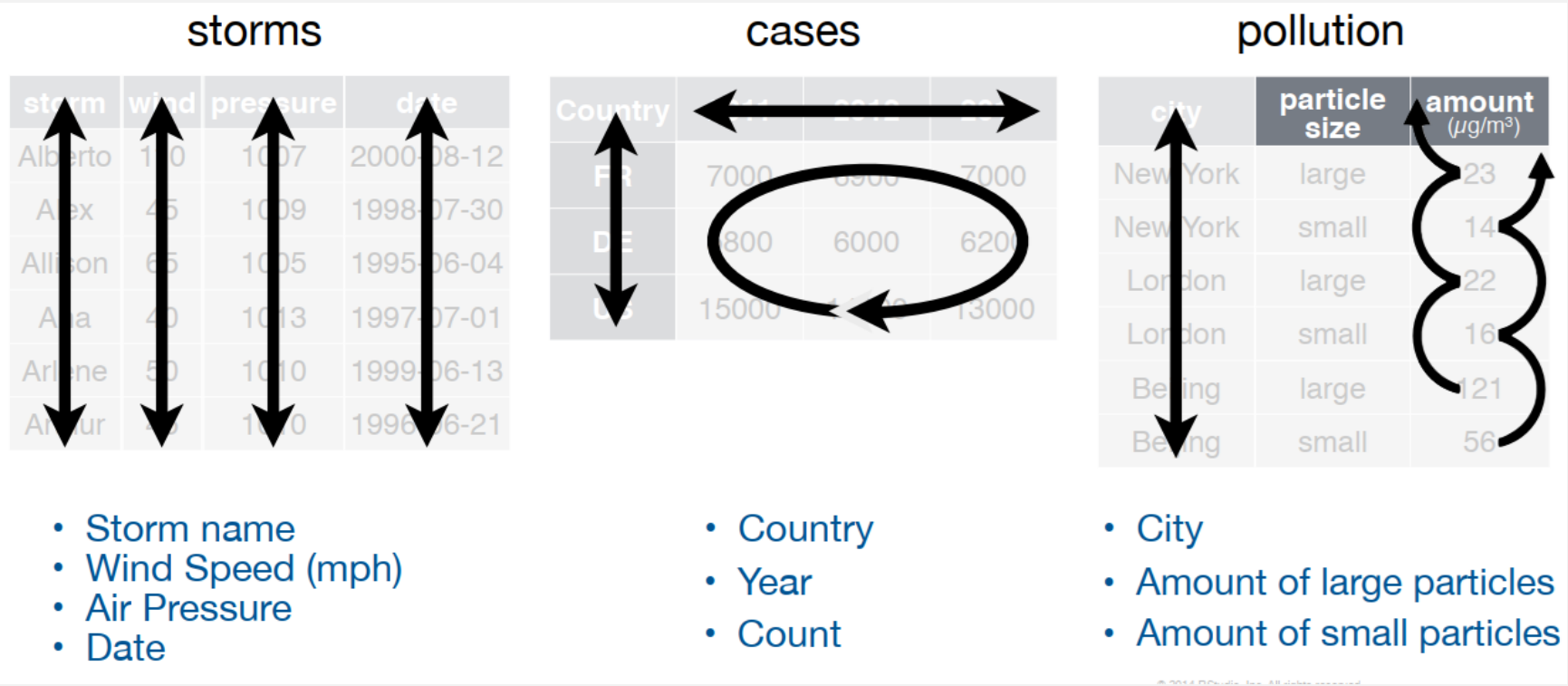
cases

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

pollution

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

TIDY DATA



- Storm name
- Wind Speed (mph)
- Air Pressure
- Date

- Country
- Year
- Count

- City
- Amount of large particles
- Amount of small particles

TIDY DATA

- Instalamos el paquete tidyr

  install.packages ("tidyr")
- Cargamos la librería

  library (tidyr)
- Funciones importantes: **gather()** y **spread()**

  ?gather

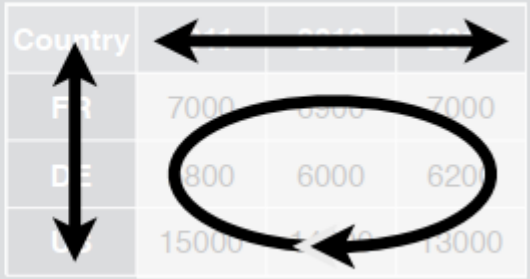
  ?spread

NUESTRO TURNO DE ORDENAR LOS DATOS

Ordenamos teniendo en cuenta 3 variables: *country, year, n.*

cases

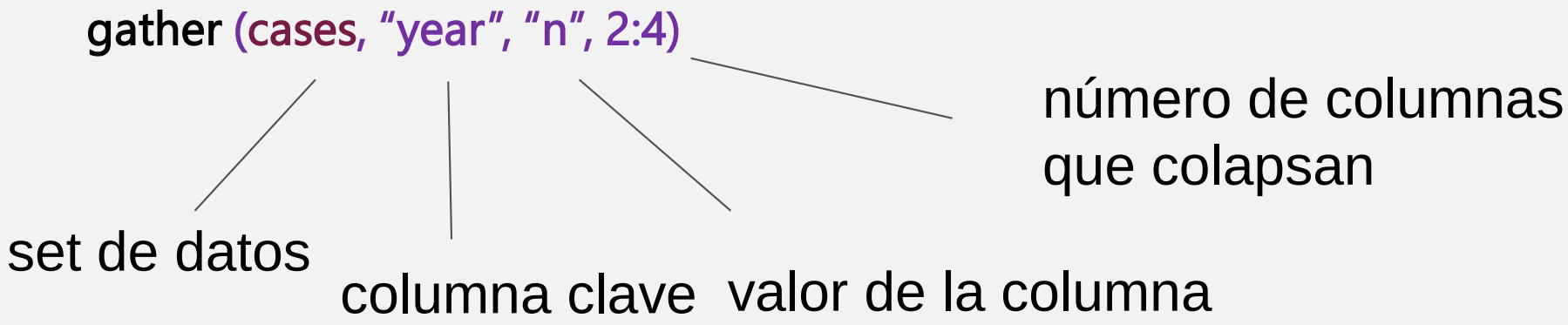
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



FUNCIÓN GATHER()

Colapsar varias columnas en una sola columna

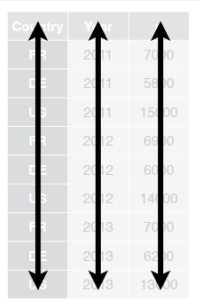
1. Una columna key que contiene los nombres de las columnas
2. Un valor que contiene los valores de las columnas.



Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

gather()

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



gather (cases, “year”, “n”, 2:4)

número de columnas que colapsan

set de datos

columna clave

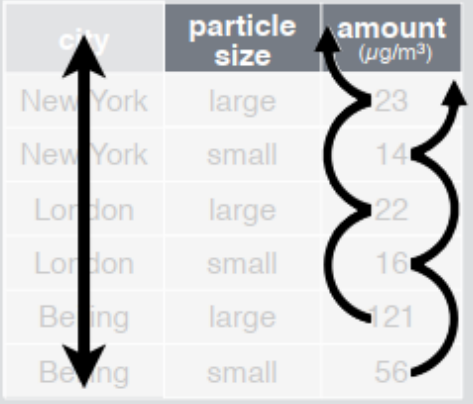
valor de la columna

NUESTRO TURNO DE ORDENAR LOS DATOS

Ordenamos teniendo en cuenta 3 variables: *city, large, small.*

pollution

city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

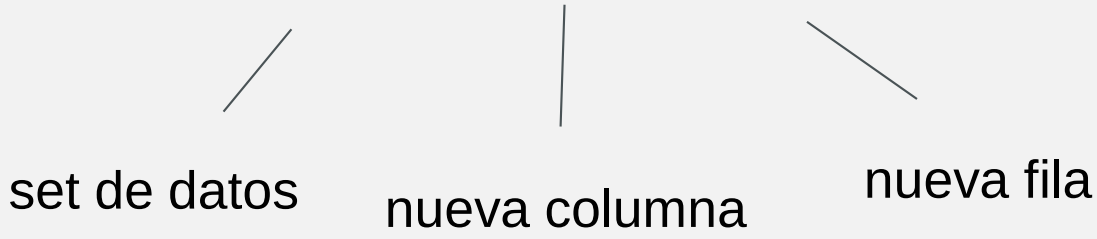


FUNCIÓN SPREAD()

Genera varias columnas a partir de dos columnas.

1. Un único valor en la columna *key* se convierte en una columna única.
2. Cada valor *value* se convierte en una fila en una nueva columna

spread (pollution, size, amount)



city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

spread()

gather()

city	large	small
New York	23	14
London	22	16
Beijing	121	56

spread (pollution, size, amount)

nueva fila

set de datos

nueva columna

FUNCIÓN SEPARATE()

Permite separar una columna en varias con un separador

```
separate(storms, date, c("year", "month", "day"), sep = "-")
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storms2

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

FUNCIÓN UNITE()

Permite unir columnas en una sola

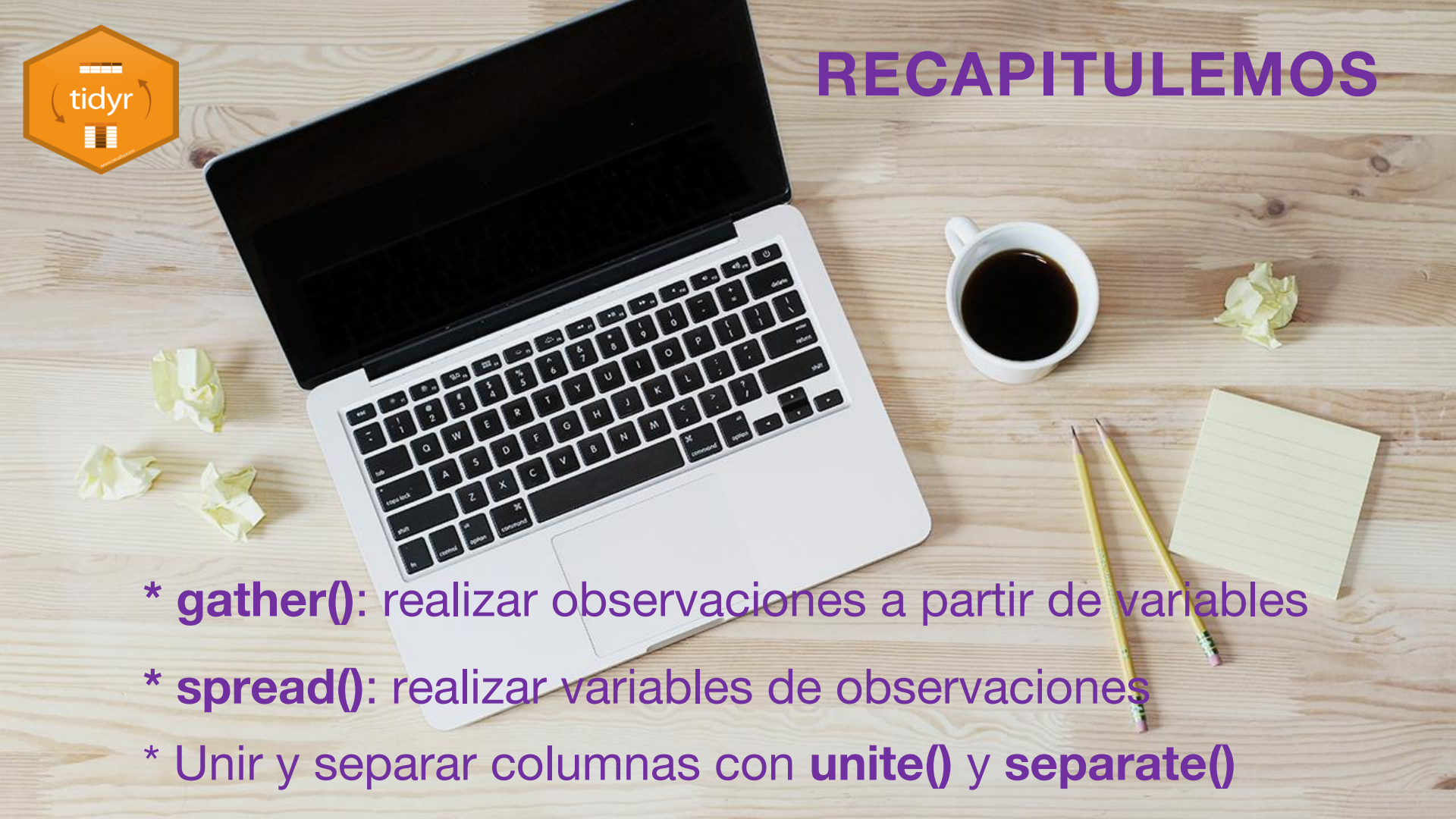
```
unite(storms2, "date", year, month, day, sep = "-")
```

storms2

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21



RECAPITULEMOS

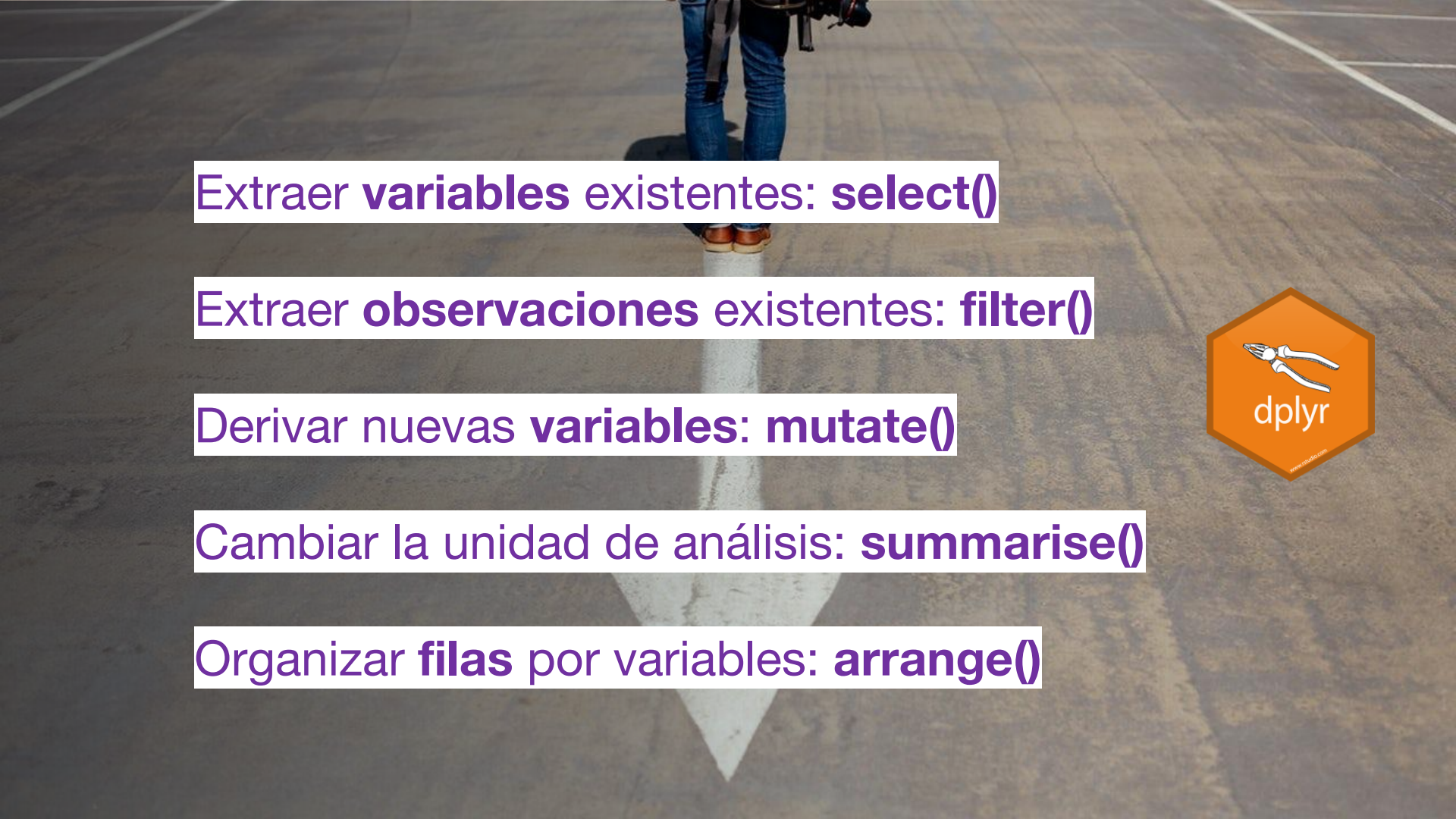
* **gather()**: realizar observaciones a partir de variables
* **spread()**: realizar variables de observaciones
* Unir y separar columnas con **unite()** y **separate()**

PARTE 3 MANIPULAR DATOS CON dplyr

DPLYR

- Paquete que nos ayuda a transformar datos tabulares.
- El paquete dplyr fue desarrollado por Hadley Wickham y es un versión optimizada de su paquete plyr.
- Proporciona una "gramática" (particularmente verbos) para la manipulación y operaciones con data frames.
- Instalación:

```
install.packages("dplyr")
library(dplyr)
install.packages("nycflights13")
library(nycflights13)
```



Extraer **variables** existentes: **select()**

Extraer **observaciones** existentes: **filter()**

Derivar nuevas **variables**: **mutate()**

Cambiar la unidad de análisis: **summarise()**

Organizar **filas** por variables: **arrange()**

Función **SELECT()**

select()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

```
select(storms, storm, pressure)
```

Función **SELECT()**

select()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

wind	pressure	date
110	1007	2000-08-12
45	1009	1998-07-30
65	1005	1995-06-04
40	1013	1997-07-01
50	1010	1999-06-13
45	1010	1996-06-21

```
select(storms, -storm)
```

Función **SELECT()**

select()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

wind	pressure	date
110	1007	2000-08-12
45	1009	1998-07-30
65	1005	1995-06-04
40	1013	1997-07-01
50	1010	1999-06-13
45	1010	1996-06-21

```
select(storms, wind:date)
```

Función **FILTER()**

filter()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13

```
filter(storms, wind >= 50)
```

Función **FILTER()**

filter()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04

```
filter(storms, wind >= 50,
  storm %in% c("Alberto", "Alex", "Allison"))
```

Función **MUTATE()**

mutate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

```
mutate(storms, ratio = pressure / wind)
```

Función **MUTATE()**

mutate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date	ratio	inverse
Alberto	110	1007	2000-08-12	9.15	0.11
Alex	45	1009	1998-07-30	22.42	0.04
Allison	65	1005	1995-06-04	15.46	0.06
Ana	40	1013	1997-07-01	25.32	0.04
Arlene	50	1010	1999-06-13	20.20	0.05
Arthur	45	1010	1996-06-21	22.44	0.04

```
mutate(storms, ratio = pressure / wind, inverse = ratio^-1)
```

Función **SUMMARISE()**

summarise()

city	particle size	amount (μg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

→

mean	sum	n
42	252	6

```
pollution %>% summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Función **ARRANGE()**

arrange()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Ana	40	1013	1997-07-01
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12

```
arrange(storms, wind)
```

Función **ARRANGE()**

arrange()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Ana	40	1013	1997-07-01
Arthur	45	1010	1996-06-21
Alex	45	1009	1998-07-30
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12

```
arrange(storms, wind, date)
```

El operador pipe %>% nos permite conectar múltiples acciones en una única “pipeline”

Con %>% podemos reescribir los comandos anteriores


```
select(storms, storm, pressure)
```

```
storms %>% select(storm, pressure)
```

```
storms %>% filter(wind >= 50)
```

```
storms %>%
  filter(wind >= 50) %>%
  select(storm, pressure)
```

El operador pipe nos permite una sintaxis clara y entendible

Empieza con un verbo

```
slice(filter(babynames,
             sex == "M"), 1)
```

```
# A tibble: 1 x 5
   year sex   name      n   prop
  <dbl> <chr> <chr> <int>  <dbl>
1  1880 M     John   9655 0.0815
```

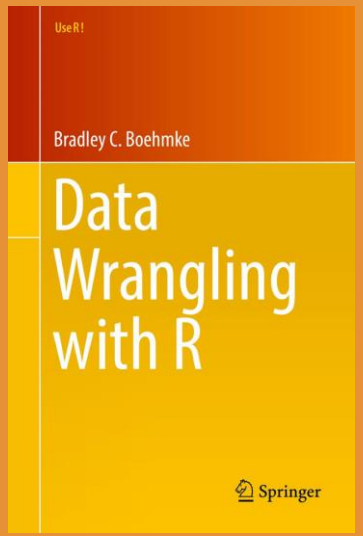
Empieza con un sustantivo (dataset) y luego la operación se indica con un verbo

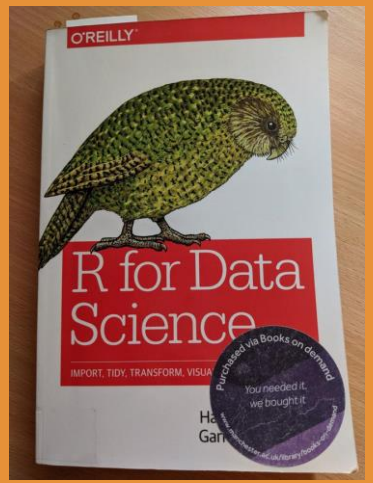
```
babynames %>%
  filter(sex == "M") %>%
  slice(1)
```

```
# A tibble: 1 x 5
   year sex   name      n   prop
  <dbl> <chr> <chr> <int>  <dbl>
1  1880 M     John   9655 0.0815
```

PARTE 4: MANIPULAR DATOS CON data.table

Fuentes de Consulta

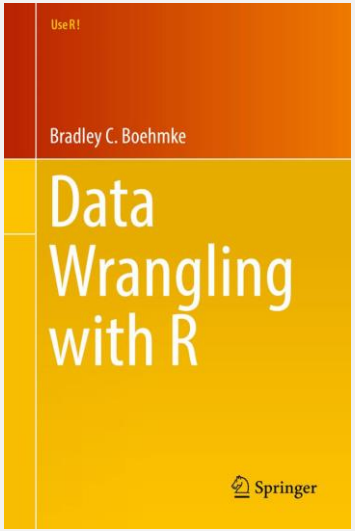


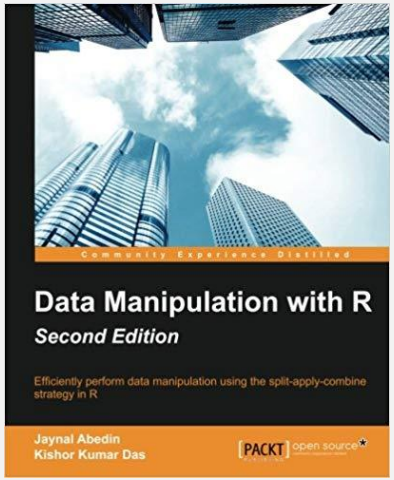


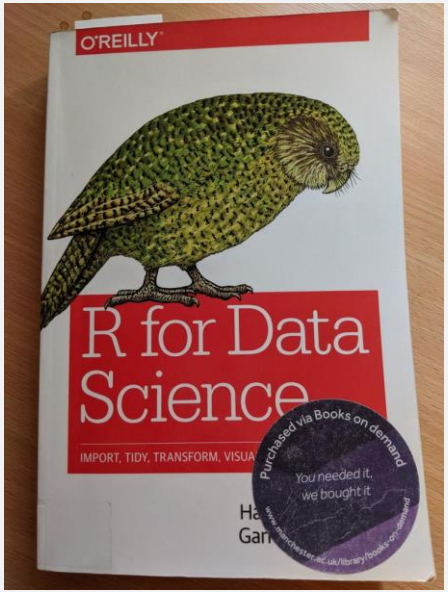
- Learning R (Github) http://bit.ly/2Aaq6d3
- R studio cheatsheets (dplyr, data.table)
- Documentación del CRAN

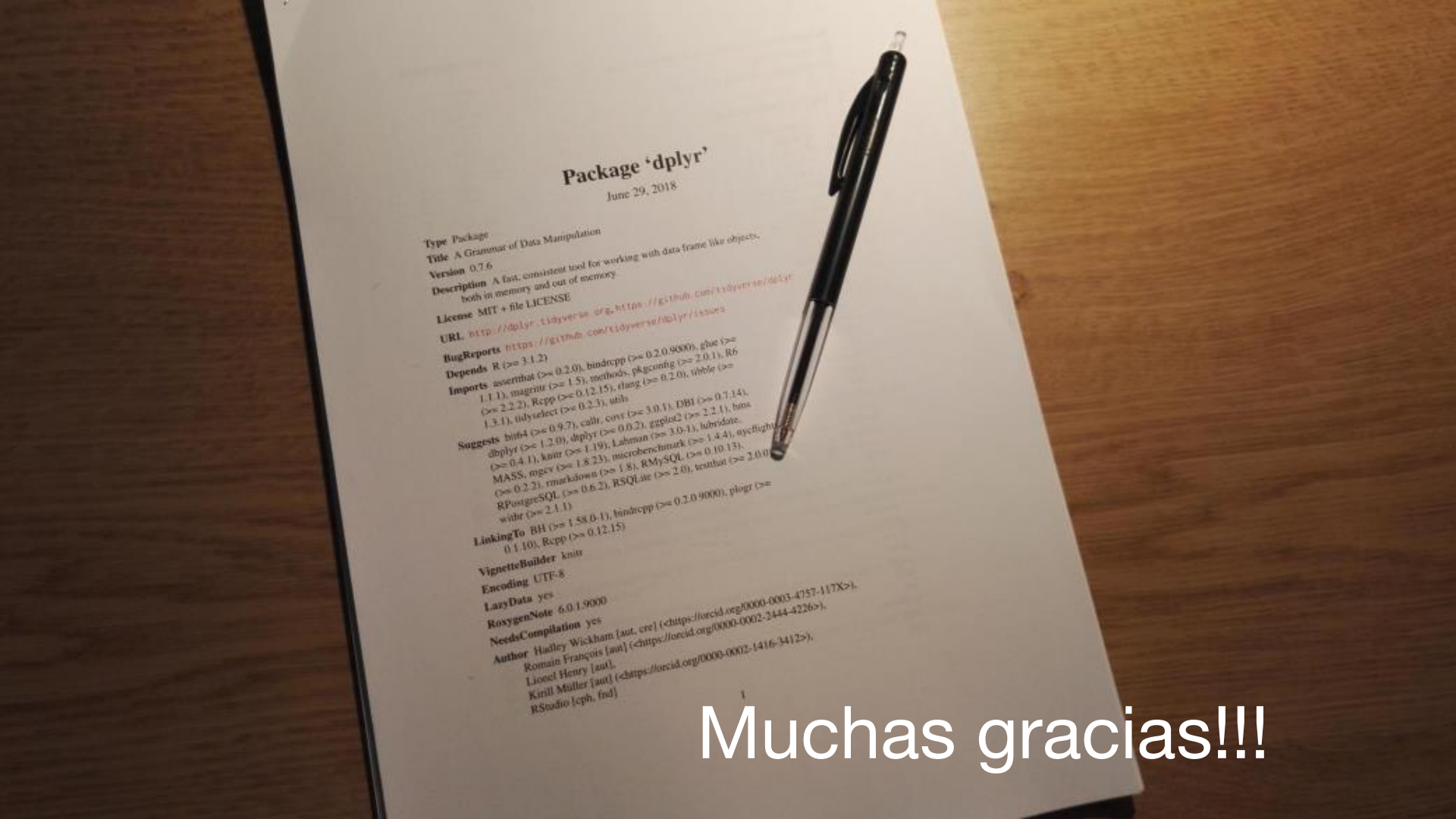
FUENTES DE CONSULTA

- Libros









Muchas gracias!!!

¡MUCHAS GRACIAS!

¿ESTAMOS EN CONTACTO?

Email: patricialoto@hotmail.com

Twitter: https://twitter.com/patriloto