

# data.table

por Mara Destéfanis



# Uso y contribución del paquete `data.table` para análisis eficiente de BigData

Mara Destéfanis

[maragdestefanis@gmail.com](mailto:maragdestefanis@gmail.com)

Descargar las diapositivas desde: [https://ml.nau.edu/latinr\\_spanish.pdf](https://ml.nau.edu/latinr_spanish.pdf)



Fundado por el programa NSF POSE, proyecto #2303612.  
diapositivas adaptadas de Toby Dylan Hocking, Arun  
Srinivasan, y datatable-viñeta de introducción - gracias!



# ¿Quien soy?

- Máster en Ciencia de Datos y Lic. Comunicación Social.
- Profesora en Colaboración con los Postgrado de Negocios de la Universidad de Belgrano Argentina y la Universidad Siglo XXI Argentina
- ¡Uso R desde el año 2021.
- Usuario de `data.table` desde 2023, contribuyo desde el año 2023.
- Consultora y Periodista Independiente.

# Resumen de la presentación

1. ¿Qué es data.table?
2. ¿Porqué data.table es tan popular/poderoso? (eficiente)
3. Usando data.table para análisis de big data eficiente.
4. Contribuyendo a data.table
  - Traducción de documentos de inglés a otro lenguaje, u\$s500.
  - Viajes de premios: u\$s 2700 para hablar de data.table en conferencias.
  - Participación en comunidad, problemas y solicitudes de extracción en GitHub.



1/4 ¿Qué es data.table?



# data.frame en R

- Estructura de datos en columnas 2D
  - filas y columnas
  - subconjunto de filas `DF[DF$id != "a", ]`
  - selección de columnas `DF[, "val"]`
  - subconjunto de filas & selección de columnas `DF[DF$id != "a", "val"]`
- Por ahí va la cosa...

DF

	<b>id</b>	<b>val</b>
1	b	4
2	a	2
3	a	3
4	c	1
5	c	5
6	b	6

2 columnas  
data.frame



# data.table

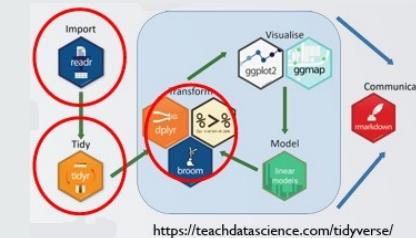
- Como data.frame, pero con más poder en la sintaxis de código R, y la implementación de código en C.
- Paquete R en CRAN desde 2006.
- Creado por Matt Dowle, co-autor Arun Srinivasan desde 2013, + 50 colaboradores.
- Hay [1463](#) paquetes en CRAN que requieren data.table (mas popular en porcentaje 0.05% de todos los paquetes de CRAN, rankea 11/19932 desde el 1 de octubre del 2023)

DF		
	id	val
1	b	4
2	a	2
3	a	3
4	c	1
5	c	5
6	b	6

2 columnas  
data.frame

# data.table y tidyverse

- El paquete de tidyverse 1.0 se publicó en **CRAN** en el año 2016.
- Los paquetes de tidyverse **tibble + readr + tidyverse + dplyr \~= data.table**
- tidyverse usa **DF|<...|>**, data.table usa **DT[...][...]**
- Tidyverse es verborrágico (tiene muchos códigos), data.table es conciso (pocos códigos) por ejemplo: tibble  
`|>filter(x=="a") |> group_by(z) |>summarise(m=mean(y))` **vs:**  
`DT\[x=="a", .(m=mean(y)), by=z\]`
- Tidyverse tiene muchas dependencias, data.table no.(es fácil de instalar)
- Tidyverse con frecuencia presenta quiebres por cambios, data.table se asegura de tener compatibilidad (para los usuarios es fácil la **actualización** hacia nuevas versiones de data.table)



2/4 ¿Porqué data.table  
es tan  
popular/poderoso?  
(eficiente)



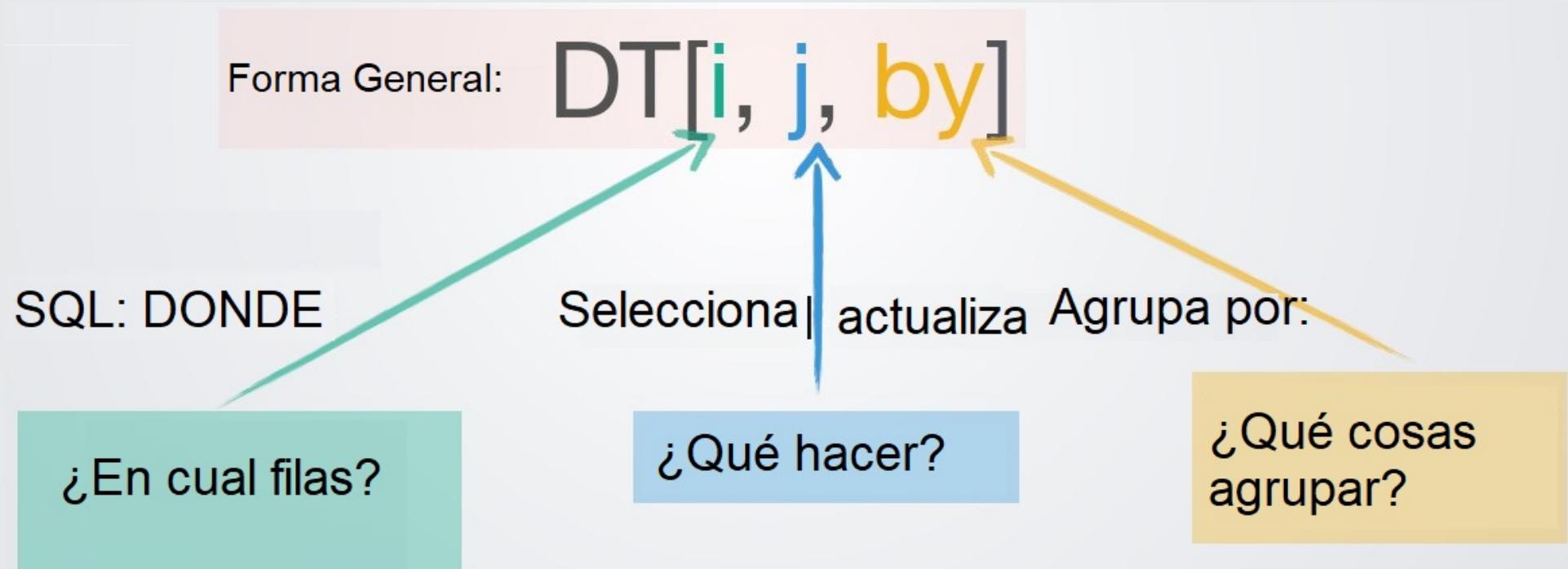
# Porque Data table tiene dos clases de eficiencia

- Eficiencia en la **sintaxis de R** (permite reducir el tiempo de programación)
- Eficiencia en la **implementación de código C** ( ya que reduce tiempo y memoria. Así un conjunto de datos puede analizarse con pocos recursos de cómputo)



# Sintaxis de R con data table

- Piensa en términos básicos de unidades
  - filas
  - columnas
  - grupo
- Matt habló en el 2014 a usuarios de R: <https://youtu.be/qLrdYhizEMg?t=1m54s>



# Sintaxis del código R con data.table

- Los datos también se estructuran en columnas.
  - 2D - filas y columnas
- subconjuntos de filas - DT[id!="a",]
- selección de columnas - DT[, val]
- computable en columnas - DT[, mean(val)]
- subconjunto de filas & selección/ computable en columnas - DT[id!="a", mean(val)]
- 3er dimensión virtual - group by

	DT	
	id	val
1:	b	4
2:	a	2
3:	a	3
4:	c	1
5:	c	5
6:	b	6

dos columnas data.table

# data.frame(DF) vs data.table(DT)

```
sum(DF[DF$code != "abd", "valA"])
```

```
DT[code != "abd", sum(valA)]
```

- Considera subconjuntos de filas con “abd” en columna de **código**, luego computa la suma de los valores en la columna **valA**.
- DF necesita que se repita, DT no lo requiere.
- **sum** puede ponerse en corchetes con DT rara vez en DF.

# data.frame(DF) vs data.table(DT)

```
DF[DF$code == "abd", "valA"] <- NA
```

```
DT[code == "abd", valA := NA]
```

- Para los subconjuntos de filas con “abd” en columna `código`, setea los valores en la columna `valA` para valores perdidos/NA
- En DF necesitamos repetir, en DT no.
- DF usa asignación de flecha `\<-`, DT usa morsa `:=`

# data.frame(DF) vs data.table(DT)

```
DF1 <- merge(DF1, DF2, all.x=TRUE)
DF1[, "valA"] <- ifelse(is.na(DF1$val),
                         DF1$valA, DF1$val)
DF1$val <- NULL
```

```
DT1[DT2, valA := val, on = .(id, code)]
```

- Resumir por grupo: considera el subgrupo de filas con “abd” en columna de **código**. Luego calcula la suma de los valores en las columnas **vaIA** y **vaIB**, para cada valor único de la columna de identificación.
- En DT todo está entre **corchetes DT []**, si usamos DF necesitamos usar otras funciones como por ejemplo: agregar **cbind**.

# data.frame(DF) vs data.table(DT)

```
DF1 <- merge(DF1, DF2, all.x=TRUE)
DF1[, "valA"] <- ifelse(is.na(DF1$val),
                         DF1$valA, DF1$val)
DF1$val <- NULL
```

```
DT1[DT2, valA := val, on = .(id, code)]
```

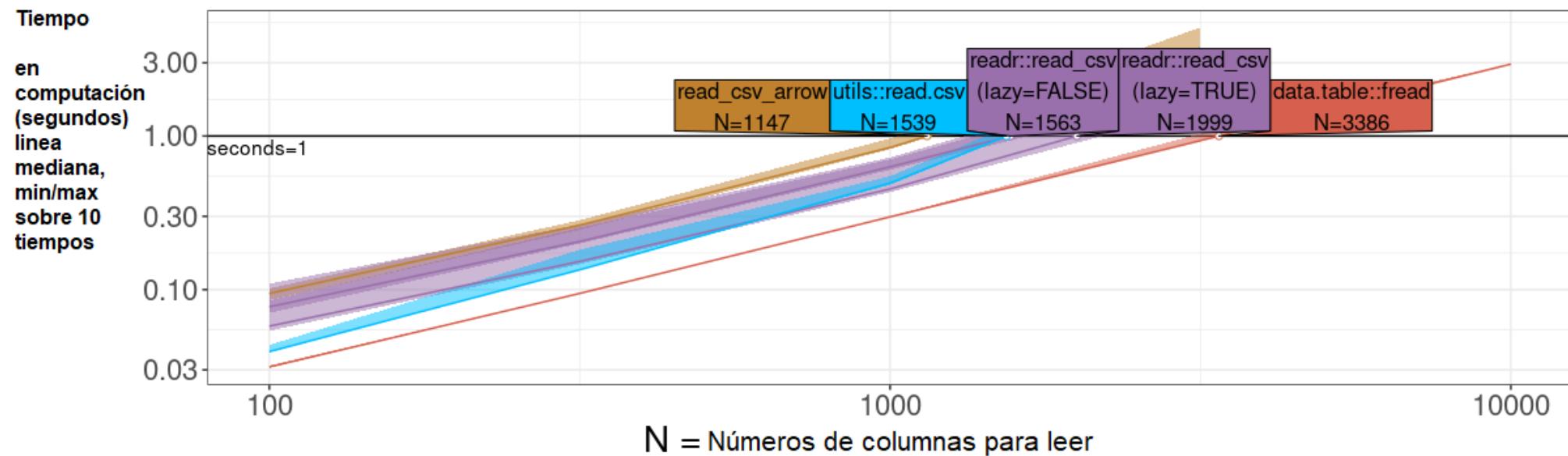
- Modificar al unirse: cada fila de DT1 que coincide con una fila de DT2 en las columnas de identificación y código, setea valA en val.
- Fácil de hacer con DT, `:=` asignación y `on=` argumento.
- Con DF en base R es posible hacer pero difícil.

# dos clases de eficiencia de data table

- Eficiencia en sintaxis de código R (permite disminuir el tiempo de programación)
- Implementando código C es eficiente ya que disminuye el tiempo y la memoria (sobre todo cuando hay largos conjuntos de datos ya que permite analizarlos usando pocos recursos de cómputo)

# `data.table::fread` es extremadamente eficiente en lector de archivos CSV

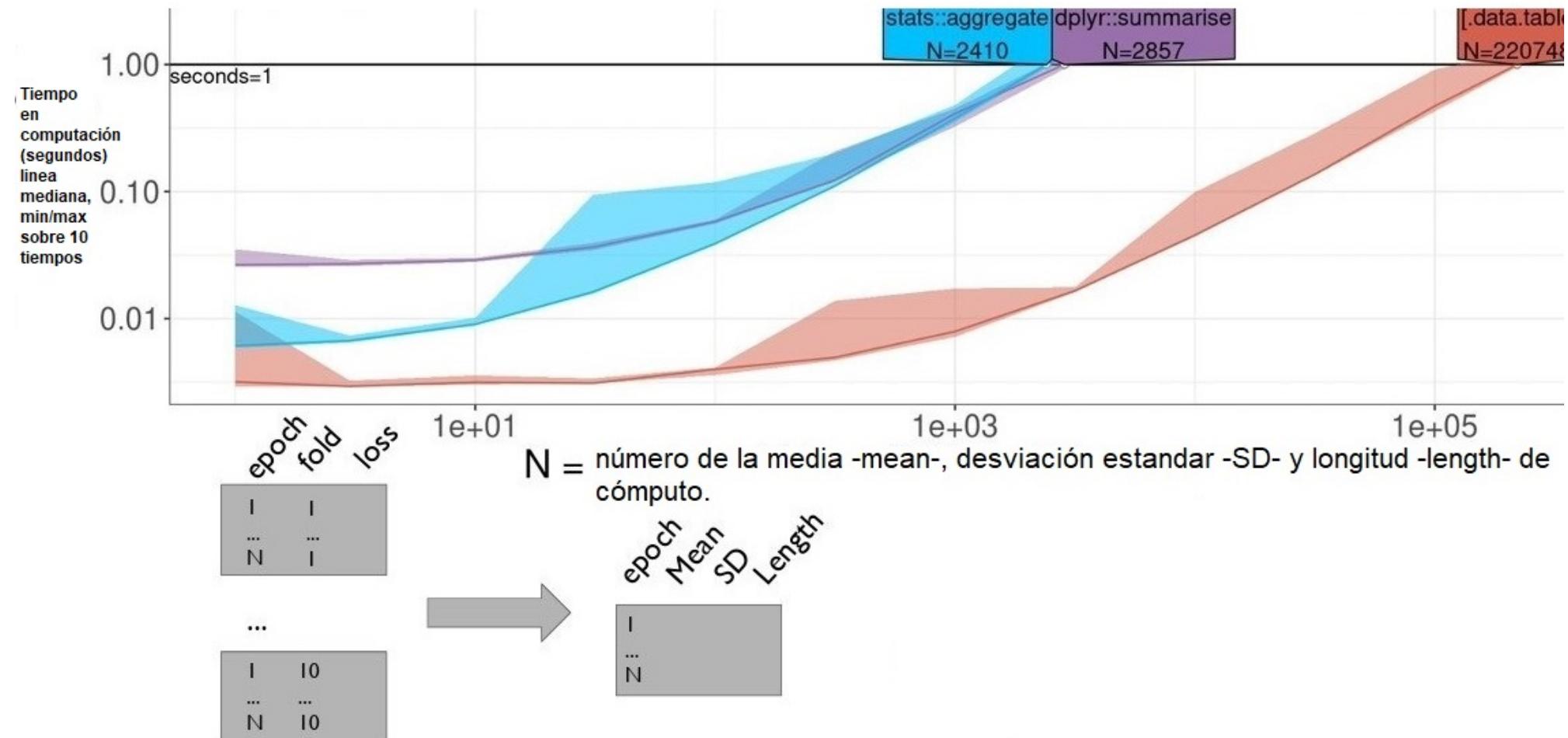
Lee números reales de CSV,  $100 \times N$



Fuente del código <https://tdhock.github.io/blog/2023/dt-atime-figures/>

`data.table` calcula resúmenes 100 veces más rápido que otros.

Promedio-mean-, desviación estandar -SD- sobre 10 números reales,  $N_{times}$

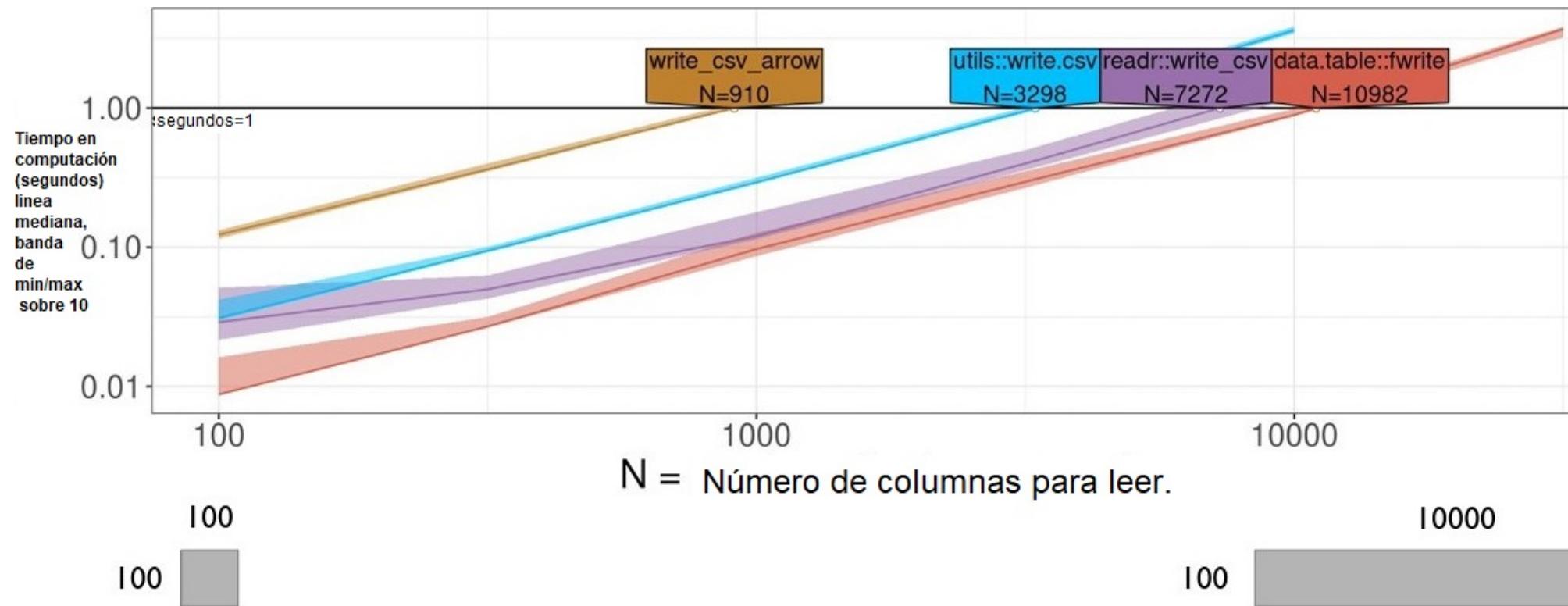


En Machine Learning, la validación cruzada k-fold estima la pérdida por hiperparámetros, como los números de epochs de entrenamiento en una red neuronal. `data.table` puede computar eficientemente el promedio de pérdida sobre  $k=10$ , por cada  $N$  epochs



## data.table::fwrite es extremadamente eficiente CSV file writer

Escribe números reales desde CSV, 100 x N



Fuente del código <https://tdhock.github.io/blog/2023/dt-atime-figures/>

# El paquete mas sobreestimado



**Conor Nash**

@conornash



Follow

Data.table is the most underrated R package. It has saved me \*days\* in waiting for analyses to complete.

Data.table es el paquete R más subestimado. Me he ahorrado **días** esperando que se completen los análisis.

# Poderoso



Alexander Flyax

@aflyax



 Follow

somebody should just write a version of [#Rstat](#)'s data.table for [#python](#). end of story. nothing as powerful exists at the moment.

Alguien debería simplemente escribir una versión de data.table de [#Rstat](#) para [#python](#), fin de la historia. No existe nada tan poderoso en este momento.

# data.table data.table



**Joey Reid**  
@JoeyPReid

# data.table



 Follow

data.table  
data.table  
data.table  
data.table  
ggplot2  
rstan  
knitr

#7FavPackages

# Gran tristeza



**Jim Savage**  
@khakieconomist



Follow

With great sadness I was forced to start using  
data.table today.

Con gran tristeza me veo hoy forzada a usar data.table.

# 3/4 Usando data.table para análisis eficiente en bigdata

# Instalando data.table, obtenemos PDF docs

- **Ejercicio:** instalar la versión de desarrollo desde GitHub, que contiene la más recientes características.
- Descargar *cheat sheet* <http://ml.nau.edu/dtcheat.pdf>
- Comenzando con R - edición tinyverse <http://ml.nau.edu/tiny.pdf>

```
>install.packages("remotes")
>remotes::install.github("Rdatatable/data.table")
(necesitas compilar: xcode para mac o rtools para windows)
o: si no tienes un compilador: instala desde CRAN la última actualización:
>install.packages("data.table")
>data.table::update_dev_pkg()
```



# Creando data.table desde R objetos

- `data.table(columna_nombre=columna_valor)`, como `data.frame`.
- valores en columnas cortas reciclados a lo largo de la columna más grande.

```
1 library(data.table)

1 data.table(
2   ID = c("b", "b", "a", "a", "c"),
3   a = 1:2,
4   b = 7,
5   c = 13:18)
```

	ID	a	b	c
	<char>	<int>	<num>	<int>
1:	b	1	7	13
2:	b	2	7	14
3:	a	1	7	15
4:	a	2	7	16
5:	c	1	7	17
6:	b	2	7	18

# data.table vs columnas frame

- Los nombres de las variables no sintácticas se mantienen de forma predeterminada.

```
1 head(data.table("10^8 m^3"Nile, year=1871:1970))
```

```
10^8 m^3  year
<ts> <int>
1:    1120 1871
2:    1160 1872
3:     963 1873
4:   1210 1874
5:   1160 1875
6:   1160 1876
```

```
1 head(data.frame("10^8 m^3"Nile, year=1871:1970))
```

```
X10.8.m.3 year
1      1120 1871
2      1160 1872
3       963 1873
4     1210 1874
5     1160 1875
6     1160 1876
```



# data.table vs columnas de frame list

```
1 data.table(L=list("scalar", c("vec","tor")),x=1:2)
```

```
      L     x
<list> <int>
1: scalar    1
2: vec,tor   2
```

```
1 data.frame(L=I(list("scalar", c("vec","tor"))),x=1:2)
```

```
      L x
1 scalar 1
2 vec, tor 2
```



# Ejercicios

- **Tu turno:** convertí el conjunto de **datos::flores** en data table vía `data.table`.
- Crea un `data.table` con una fila, y columnas con los siguientes datos: tu nombre (*character*), años de experiencia con R (*numeric*), lugar de nacimiento(*character*), ciudad de residencia(*character*), idiomas(s) (*list of character*)

# Lee CSV desde el archivo

- Vuelos que despegaron de Nueva York (JFK, LGA o EWR) durante 2013. datos::vuelos

```
if(!file.exists("vuelos.csv"))download.file(  
  "https://ml.nau.edu/vuelos.csv", "vuelos.csv")  
(vuelos<-data.table::fread("vuelos.csv"))
```

# Lee CSV desde el comando shell

- Es útil con comandos que generan datos de salida en CSV ( no archivos intermedios)
- *egrep* es útil si no necesitas/o querés leer todo el conjunto de datos en R

```
data.table::fread("egrep 'LAX|anio' vuelos.csv")
```

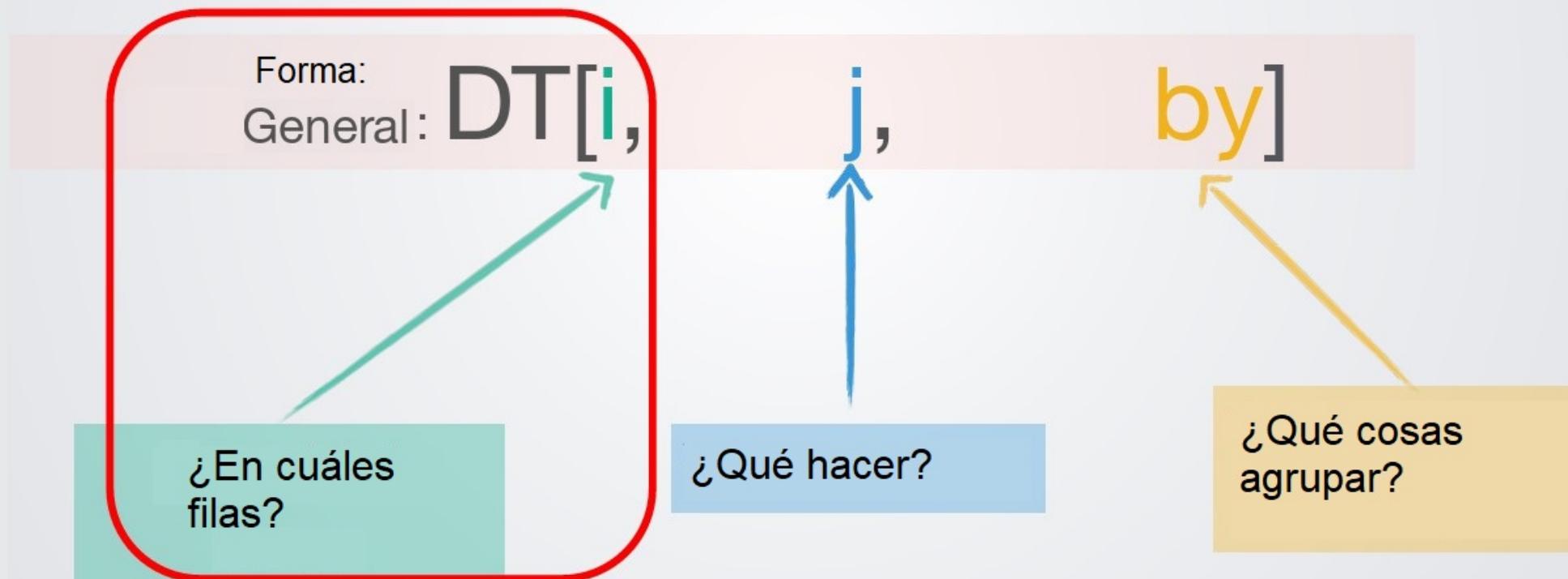
# Ejercicio: Lee CSV desde URL

- Usualmente puede descargarse el archivo primero `download.file()`, para evitar tener que bajar de nuevo el archivo *re-starting R* Es conveniente.
- Llegó tu turno. Trata con el siguiente código:

```
vuelos <- data.table::fread("http://ml.nau.edu/vuelos.csv")
```

# Subconjunto de filas usando i

- Piensa en términos básicos de unidades
  - filas
  - columnas
  - grupo
- La sintaxis de data.table provee marcador de posición para cada uno de ellos.



# Subconjunto de filas lógicas

- Selecciona todos los vuelos con “JFK” del aeropuerto de origen en el mes de Junio.
- DT[rows, ] la coma no es necesaria (pero permitida).

```
1 vuelos[origen == "JFK" & mes == 6]

    anio   mes   dia horario_salida salida_programada atraso_salida
    <int> <int> <int>           <int>           <int>           <num>
 1: 2013     6     1             2            2359            3
 2: 2013     6     1            538            545           -7
 3: 2013     6     1            539            540           -1
 4: 2013     6     1            553            600           -7
 5: 2013     6     1            554            600           -6
---
9468: 2013     6    30            NA            2000            NA
9469: 2013     6    30            NA            1725            NA
9470: 2013     6    30            NA            2045            NA
9471: 2013     6    30            NA            1405            NA
9472: 2013     6    30            NA            1945            NA
    horario_llegada llegada_programada atraso_llegada aerolinea vuelo
    <int>           <int>           <num>       <char> <int>
 1:        341          350           -9         B6    739
 2:        925          922            3         B6    725
 3:        832          840           -8         AA    701
 4:        700          711           -11        EV    5716
 5:        851          908           -17        UA   1159
---
9468:          NA          2127            NA      9E  4287
```



# Conjunto de filas enteras

- Toma los primeros dos vuelos, o todos exceptuando el primero de los dos.
- Conjunto de enteros (*Integer*) en DT trabaja como en DF.

```
1 vuelos[1:2]
```

	anio	mes	dia	horario_salida	salida_programada	atraso_salida
	<int>	<int>	<int>	<int>	<int>	<num>
1:	2013	1	1	517	515	2
2:	2013	1	1	533	529	4

```
horario_llegada llegada_programada atraso_llegada aerolinea vuelo
<int> <int> <num> <char> <int>
1: 830 819 11 UA 1545
2: 850 830 20 UA 1714
```

```
codigo_cola origen destino tiempo_vuelo distancia hora minuto
<char> <char> <char> <num> <num> <num> <num>
1: N14228 EWR IAH 227 1400 5 15
2: N24211 LGA IAH 227 1416 5 29
```

```
fecha_hora
<POSc>
```

```
1: 2013-01-01 05:00:00
2: 2013-01-01 05:00:00
```

```
1 vuelos[-(1:2)]
```

	anio	mes	dia	horario_salida	salida_programada	atraso_salida
	<int>	<int>	<int>	<int>	<int>	<num>
1:	2013	1	1	542	540	2
2:	2013	1	1	544	545	-1
3:	2013	1	1	554	600	-6
4:	2013	1	1	554	558	-4
5:	2013	1	1	555	600	-5

---



336770:	2013	9	30	NA	1455	NA
336771:	2013	9	30	NA	2200	NA
336772:	2013	9	30	NA	1210	NA
336773:	2013	9	30	NA	1159	NA
336774:	2013	9	30	NA	840	NA
				horario_llegada	llegada_programada	atraso_llegada
				aerolinea	vuelo	
				<char>	<int>	
1:		923		AA	850	1141
2:		1004		B6	1022	725
3:		812		DL	837	461
4:		740		UA	728	1696
5:		913		B6	854	507
---						
336770:		NA	1634	NA	9E	3393



# Ordenar filas

- Ordena los vuelos primero por columna original en orden ascendente, y luego por destinos en orden descendentes.
- Ordena data.table de manera eficiente que se integre a la base R en 2016!

```
1 vuelos[order(origen, -destino)]
```

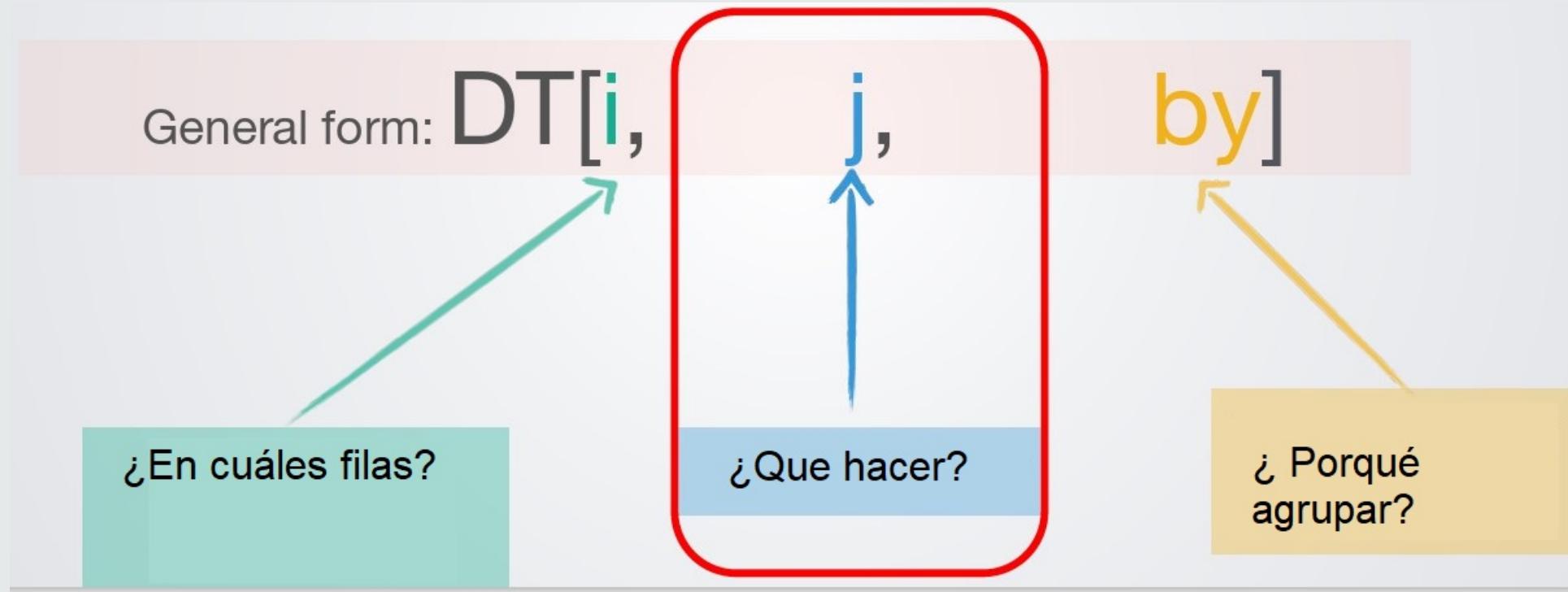
	anio	mes	dia	horario_salida	salida_programada	atraso_salida
	<int>	<int>	<int>	<int>	<int>	<num>
1:	2013	1	2	905	822	43
2:	2013	1	3	848	850	-2
3:	2013	1	4	901	850	11
4:	2013	1	6	843	848	-5
5:	2013	1	7	858	850	8
---						
336772:	2013	9	30	1752	1759	-7
336773:	2013	9	30	1851	1900	-9
336774:	2013	9	30	1903	1859	4
336775:	2013	9	30	1944	1950	-6
336776:	2013	9	30	1955	2000	-5
---						
336772:	2021			2031	-10	DL 61
	horario_llegada	llegada_programada	atraso_llegada	aerolinea	vuelo	
	<int>	<int>	<num>	<char>	<int>	
1:	1313	1045	NA	EV	4140	
2:	1149	1113	36	EV	4125	
3:	1120	1113	7	EV	4125	
4:	1053	1111	-18	EV	4625	
5:	1105	1113	-8	EV	4125	
---						

# Ejercicios

- Tu turno. ¿Cuantas filas hay con aerolinea “AA” y destino “LAX”?
- Si ordenas la tabla. ¿Cuál es la primer fecha y ultima mes/dia/hora?
- Una pista: puedes usar DT[...][...] en una cadena.  
Similar para *piping*, DF |>...|>...

# Computa columnas in j

- Piensa en términos básicos de unidades
  - filas
  - columnas
  - grupo
- La sintaxis de data.table provee Marcador de posición para cada una de ellas.



# Selecciona una columna

- DT[,“arr\_delay”] -> data.table con una columna.
- DT[,“arr\_delay”] -> vector  
como DT\$arr\_delay o DT[["arr\_delay"]] en base a R
- **Ejercicio:** usa hist() para dibujar un histograma deatrás de llegadas

```
1 str(vuelos[, "atraso_llegada"])

Classes 'data.table' and 'data.frame': 336776 obs. of 1 variable:
$ atraso_llegada: num 11 20 33 -18 -25 12 19 -14 -8 8 ...
- attr(*, ".internal.selfref")=<externalptr>

1 str(vuelos[, atraso_llegada]) # como DT[["atraso_llegada"]] o DT$retraso_llegada
num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
```

# Selecciona 2 columnas con literal

Selecciona dos columnas con  
variable

- Puedes usar con `=FALSE` o **doble puntos** ...
 

( como un nivel arriba en la ruta del archivo *unix*, busca `select_cols` no como un nombre de columna)

```
::: {.cell}

```{.r .cell-code}
(DT=data.table(x=1:2, y=3:4,
  select_cols=c("foo", "bar")))
```

::: {.cell-output .cell-output-stdout}
```
      x     y select_cols
<int> <int>      <char>
1:     1     3          foo
2:     2     4          bar
```

:::
:::

::: {.cell}

```{.r .cell-code}
select_cols=c("x", "y")
DT[, select_cols]
1 DT[,...select_cols]
```
      x     y
<int> <int>
1:     1     3
2:     2     4

1 DT[, select_cols, with=FALSE]
```
      x     y
<int> <int>
1:     1     3
2:     2     4
```

```

# Selecciona todas las columnas excepto dos

- Puedes usar ambos! o para negar la selección de columnas.

```

1 retrasos_cols= c("atraso_llegada", "atraso_salida")
1 head(vuelos[, !..retrasos_cols])

    anio   mes   dia horario_salida salida_programada horario_llegada
    <int> <int> <int>           <int>           <int>           <int>
1: 2013     1     1           517           515           830
2: 2013     1     1           533           529           850
3: 2013     1     1           542           540           923
4: 2013     1     1           544           545          1004
5: 2013     1     1           554           600           812
6: 2013     1     1           554           558           740
llegada_programada aerolinea vuelo codigo_cola origen destino tiempo_vuelo
    <int>      <char> <int>      <char> <char> <char>       <num>
1:           819      UA  1545    N14228    EWR    IAH      227
2:           830      UA  1714    N24211    LGA    IAH      227
3:           850      AA  1141    N619AA    JFK    MIA      160
4:          1022      B6   725    N804JB    JFK    BQN      183
5:           837      DL   461    N668DN    LGA    ATL      116
6:           728      UA  1696    N39463    EWR    ORD      150
distancia hora minuto                fecha_hora
    <num> <num> <num>                  <POSc>
1:    1400     5    15 2013-01-01 05:00:00
2:    1416     5    29 2013-01-01 05:00:00
3:    1089     5    40 2013-01-01 05:00:00
4:    1576     5    45 2013-01-01 05:00:00

```

```
1 head(vuelos[, -..retrasos_cols])
```

|    | anio               | mes       | dia    | horario_salida      | salida_programada | horario_llegada |              |
|----|--------------------|-----------|--------|---------------------|-------------------|-----------------|--------------|
|    | <int>              | <int>     | <int>  | <int>               | <int>             | <int>           |              |
| 1: | 2013               | 1         | 1      | 517                 | 515               | 830             |              |
| 2: | 2013               | 1         | 1      | 533                 | 529               | 850             |              |
| 3: | 2013               | 1         | 1      | 542                 | 540               | 923             |              |
| 4: | 2013               | 1         | 1      | 544                 | 545               | 1004            |              |
| 5: | 2013               | 1         | 1      | 554                 | 600               | 812             |              |
| 6: | 2013               | 1         | 1      | 554                 | 558               | 740             |              |
|    | llegada_programada | aerolinea | vuelo  | codigo_colas        | origen            | destino         | tiempo_vuelo |
|    | <int>              | <char>    | <int>  | <char>              | <char>            | <char>          | <num>        |
| 1: | 819                | UA        | 1545   | N14228              | EWR               | IAH             | 227          |
| 2: | 830                | UA        | 1714   | N24211              | LGA               | IAH             | 227          |
| 3: | 850                | AA        | 1141   | N619AA              | JFK               | MIA             | 160          |
| 4: | 1022               | B6        | 725    | N804JB              | JFK               | BQN             | 183          |
| 5: | 837                | DL        | 461    | N668DN              | LGA               | ATL             | 116          |
| 6: | 728                | UA        | 1696   | N39463              | EWR               | ORD             | 150          |
|    | distancia          | hora      | minuto | fecha_hora          |                   |                 |              |
|    | <num>              | <num>     | <num>  | <POS>c              |                   |                 |              |
| 1: | 1400               | 5         | 15     | 2013-01-01 05:00:00 |                   |                 |              |
| 2: | 1416               | 5         | 29     | 2013-01-01 05:00:00 |                   |                 |              |
| 3: | 1089               | 5         | 40     | 2013-01-01 05:00:00 |                   |                 |              |
| 4: | 1576               | 5         | 45     | 2013-01-01 05:00:00 |                   |                 |              |



# Selecciona rangos de columnas

- Puedes usar columna1:columna2 para seleccionar ambas, o todas entre ellas(y puedes usar negativo con ! o -)

```
1 head(vuelos[, anio:dia])
```

|    | anio  | mes   | dia   |
|----|-------|-------|-------|
|    | <int> | <int> | <int> |
| 1: | 2013  | 1     | 1     |
| 2: | 2013  | 1     | 1     |
| 3: | 2013  | 1     | 1     |
| 4: | 2013  | 1     | 1     |
| 5: | 2013  | 1     | 1     |
| 6: | 2013  | 1     | 1     |

```
1 head(vuelos[, !(anio:dia)])
```

|    | horario_salida | salida_programada | atraso_salida | horario_llegada |
|----|----------------|-------------------|---------------|-----------------|
|    | <int>          | <int>             | <num>         | <int>           |
| 1: | 517            | 515               | 2             | 830             |
| 2: | 533            | 529               | 4             | 850             |
| 3: | 542            | 540               | 2             | 923             |
| 4: | 544            | 545               | -1            | 1004            |
| 5: | 554            | 600               | -6            | 812             |



|        | 554                | 558            | -4        | 740    |            |               |
|--------|--------------------|----------------|-----------|--------|------------|---------------|
| 6:     | llegada_programada | atraso_llegada | aerolinea | vuelo  | codigo_col | origen        |
|        | destino            | <int>          | <num>     | <char> | <int>      | <char> <char> |
| <char> |                    |                |           |        |            |               |
| 1:     |                    | 819            | 11        | UA     | 1545       | N14228 EWR    |
| IAH    |                    |                |           |        |            |               |
| 2:     |                    | 830            | 20        | UA     | 1714       | N24211 LGA    |
| ---    |                    |                |           |        |            |               |

# Renombra y computa las columnas

- **Ejercicio:** trata de usar `.()` o `list()` en vez de `data.table()` en `j`
- `.()` es un alias de `list()-items` en lista son columnas de la tabla.

```
1 vuelos[, data.table(puntodestino=destino, aire_horas=tiempo_vuelo/60)]
```

```
puntodestino    aire_horas
<char>        <num>
1:      IAH     3.783333
2:      IAH     3.783333
3:      MIA     2.666667
4:      BQN     3.050000
5:      ATL     1.933333
---
336772:      DCA       NA
336773:      SYR       NA
336774:      BNA       NA
336775:      CLE       NA
336776:      RDU       NA
```

# Resumir todas las filas

```
1 vuelos[, .(m_arr = mean(atraso_llegada), m_dep = mean(atraso_salida))]
```

```
m_arr m_dep
<num> <num>
1:     NA     NA
```

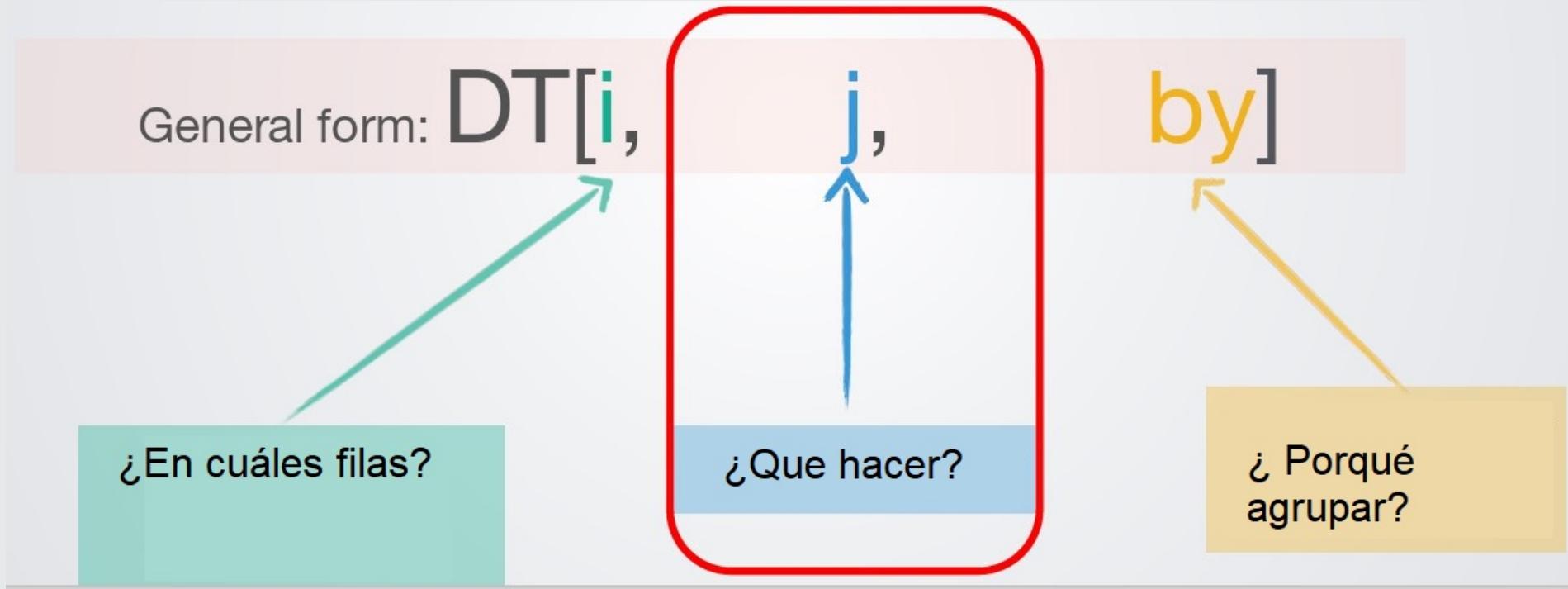
```
1 vuelos[, .(rango_llegada = range(atraso_llegada), stat=c("min","max"))]
```

```
rango_llegada    stat
<num> <char>
1:          NA   min
2:          NA   max
```

- En `data.table/list/.en j`, resume funciones que pueden usarse.
- En este caso podemos usar `mean`, con retorno de un número simple.
- Podemos también usar `range`, con la cual retorna dos números.
- Podemos computar `mean min max` de arribos y partidas de todos los vuelos.
- Ejercicio: computa min/max `tiempo_vuelo` y `distancia`

# Crea/elimina columnas en j

- Piensa en términos básicos de unidades -  
**filas.**  
**columnas.**  
**grupo.**
- Podemos usar DT[, variable:= valor] o set(DT, j="variable", valor=valor)



# Comparando columnas en base a R en `data.table`

- En la columna denominada “mundo” establece el valor de “hola”
- Es posible utilizar código base R con `data.table`.
- Históricamente, `:=` ha sido más eficiente en (tiempo/memoria)
- Recientes versiones de R lo han mejorado enormemente!

|          | Base código R                               | código <code>data.table</code>                     |
|----------|---|--|
| Literal  | <code>DF\$mundo &lt;- "Hola"</code>         | <code>DT[, mundo := "hola" ]</code>                |
| Variable | <code>DF[mi_var_nombre] &lt;- "Hola"</code> | <code>set(DT,j=mi_var_nombre, value="Hola")</code> |

# Crear valor en las columnas :=

- DT[, variable := valor] puede usarse para crear nuevas columnas o actualizaciones.
- Mismo efecto que DT\$variable <- valor, o DT[["variable"]] <- valor

```
1 mes.dt <- vuelos[, .(mes, tiempo_vuelo, distancia)]
```

```
1 head(mes.dt)
```

|    | mes   | tiempo_vuelo | distancia |
|----|-------|--------------|-----------|
|    | <int> | <num>        | <num>     |
| 1: | 1     | 227          | 1400      |
| 2: | 1     | 227          | 1416      |
| 3: | 1     | 160          | 1089      |
| 4: | 1     | 183          | 1576      |
| 5: | 1     | 116          | 762       |
| 6: | 1     | 150          | 719       |

```
1 mes.dt[, aire_horas := tiempo_vuelo/60]
```

```
1 head(mes.dt)
```

|    | mes   | tiempo_vuelo | distancia | aire_horas |
|----|-------|--------------|-----------|------------|
|    | <int> | <num>        | <num>     | <num>      |
| 1: | 1     | 227          | 1400      | 3.783333   |
| 2: | 1     | 227          | 1416      | 3.783333   |
| 3: | 1     | 160          | 1089      | 2.666667   |
| 4: | 1     | 183          | 1576      | 3.050000   |
| 5: | 1     | 116          | 762       | 1.933333   |
| 6: | 1     | 150          | 719       | 2.500000   |

# Elimina columnas := NULL

- DT[, variable :=NULL] elimina columnas
- Todas := las operaciones son por referencias (eficiente)

```
1 head(mes.dt)

mes tiempo_vuelo distancia aire_horas
<int>      <num>     <num>      <num>
1:    1        227     1400  3.783333
2:    1        227     1416  3.783333
3:    1        160     1089  2.666667
4:    1        183     1576  3.050000
5:    1        116      762  1.933333
6:    1        150      719  2.500000
```

```
1 mes.dt[, tiempo_vuelo := NULL]
```

```
1 head(mes.dt)

mes distancia aire_horas
<int>      <num>      <num>
1:    1       1400  3.783333
2:    1       1416  3.783333
3:    1       1089  2.666667
4:    1       1576  3.050000
5:    1       762   1.933333
6:    1       719   2.500000
```

# set() para crear/eliminar columnas

- set(DT, j="col\_name", value=col\_val) es igual que DT[, col\_name :=col\_val]
- Usualmente cuando los nombres de las columnas esta almacenado en una variable, como aquí:

```

1 my_var_name = "nuevita"
2 set(vuelos, j=my_var_name, value="HOLA!")
3 vuelos

Index: <mes_origen>
      anio   mes   dia horario_salida salida_programada atraso_salida
      <int> <int> <int>          <int>           <int>           <num>
1: 2013     1     1            517            515             2
2: 2013     1     1            533            529             4
3: 2013     1     1            542            540             2
4: 2013     1     1            544            545            -1
5: 2013     1     1            554            600            -6
---
336772: 2013     9    30            NA           1455            NA
336773: 2013     9    30            NA           2200            NA
336774: 2013     9    30            NA           1210            NA
336775: 2013     9    30            NA           1159            NA
336776: 2013     9    30            NA           840            NA
      horario_llegada llegada_programada atraso_llegada aerolinea vuelo
      <int>           <int>           <num>           <char> <int>
1:         830            819            11            UA  1545
2:         850            830            20            UA  1714

```



```

3:          923          850          33      AA 1141
4:          1004         1022         -18      B6  725
5:          812          837         -25      DL  461
---
```

```

1 set(vuelos, j=my_var_name, value=NULL)
2 vuelos
```

Index: <mes\_origen>

|    | anio  | mes   | dia   | horario_salida | salida_programada | atraso_salida |
|----|-------|-------|-------|----------------|-------------------|---------------|
|    | <int> | <int> | <int> | <int>          | <int>             | <num>         |
| 1: | 2013  | 1     | 1     | 517            | 515               | 2             |
| 2: | 2013  | 1     | 1     | 533            | 529               | 4             |
| 3: | 2013  | 1     | 1     | 542            | 540               | 2             |
| 4: | 2013  | 1     | 1     | 544            | 545               | -1            |
| 5: | 2013  | 1     | 1     | 554            | 600               | -6            |

---

|         |      |   |    |    |      |    |
|---------|------|---|----|----|------|----|
| 336772: | 2013 | 9 | 30 | NA | 1455 | NA |
| 336773: | 2013 | 9 | 30 | NA | 2200 | NA |
| 336774: | 2013 | 9 | 30 | NA | 1210 | NA |
| 336775: | 2013 | 9 | 30 | NA | 1159 | NA |
| 336776: | 2013 | 9 | 30 | NA | 840  | NA |

horario\_llegada llegada\_programada atraso\_llegada aerolinea vuelo

|    | horario_llegada | llegada_programada | atraso_llegada | aerolinea | vuelo |
|----|-----------------|--------------------|----------------|-----------|-------|
|    | <int>           | <int>              | <num>          | <char>    | <int> |
| 1: | 830             | 819                | 11             | UA        | 1545  |
| 2: | 850             | 830                | 20             | UA        | 1714  |
| 3: | 923             | 850                | 33             | AA        | 1141  |
| 4: | 1004            | 1022               | -18            | B6        | 725   |
| 5: | 812             | 837                | -25            | DL        | 461   |

---



# Ejercicios

- Crea nuevas columnas aire\_horas := tiempo\_aire/60 como en las diapositivas previas.
- Crea una nueva columna hora\_llegada := + aire\_horas
- Usa vuelos[, range(horas)] para encontrar el rango normal de la variable hora.
- Hace un subconjunto para encontrar cuantas filas tienen hora\_llegada > max(hora)
- Crea una nueva variable dia\_llegada con la cual sea “same” or “next”  
ifelse(hora\_llegada>max(hour), “next”, “same”)



Computa un subconjunto con i  
y j

# Resumiendo un subconjunto de filas

```

1 vuelos[origen == "JFK" & mes == 6L, length(destino) ]
[1] 9472
1 vuelos[origen == "JFK" & mes == 6L, .N]
[1] 9472
1 vuelos[origen == "JFK" & mes == 6L, .(m_llegada=mean(atraso_llegada), num_vuelos=.N) ]
m_llegada num_vuelos
<num>      <int>
1:       NA      9472

```

- ¿Cuántos viajes se realizaron desde el año 2014 desde el aeropuerto de “JFK” en el mes de junio y cual es el promedio de partidas y retraso en la llegada ?
- Especial símbolo .N puede usarse en [] para tener los números de las filas.
- Ejercicio: para todos los vuelos con positivas partidas, ¿Cuál fue el promedio *mean* de demora en los arribos?y ¿Para vuelos con negativas partidas?

# Actualización de columnas

## con :=

- DT[ i , variable := value] se puede utilizar para actualizar filas que coinciden con i
- set(DT, i= , j="variable", value= ) útil para programar

```

1 head(mes.dt)

  mes distancia aire_horas
  <int>    <num>      <num>
1:   1        1400    3.783333
2:   1        1416    3.783333
3:   1        1089    2.666667
4:   1        1576    3.050000
5:   1        762     1.933333
6:   1        719     2.500000

1 mes.dt[distancia<100, aire_horas := 0]

1 head(mes.dt)

  mes distancia aire_horas
  <int>    <num>      <num>
1:   1        1400    3.783333
2:   1        1416    3.783333
3:   1        1089    2.666667
4:   1        1576    3.050000
5:   1        762     1.933333
6:   1        719     2.500000

```



# Ejercicios

- Continúa de previos ejercicios...
- Crea nuevas columnas `horas_aire := tiempo_vuelo/60` como las diapositivas previas.
- crea una nueva columna `horas_llegada := hora + horas_aire`.
- Crea una nueva columna `dia_llegada` de la cual sea ambos iguales o siguientes  
`ifelse(horas_llegada>max(hora), "next", "same")`
- Actualiza `horas_llegada := horas_llegadas-24` para las filas con `horas_llegadas > 24`.
- Usa `vuelos[, range(horas_llegada)]` para confirmar el valor entre 0 y 24.

Computa por c/ grupo con j y  
by

# Resumiendo por cada grupo

- `by=origen` significa tratar a cada único valor de origen como un grupo.
- Por cada grupo/origen, podemos computar el número de vuelos y el promedio *mean* de los arribos y partidas.

```
1 vuelos[, .(num_vuelos=.N), by=origen]
```

```
origen num_vuelos
<char>      <int>
1:   EWR      120835
2:   LGA      104662
3:   JFK      111279
```

```
1 vuelos[, .(num_vuelos=.N, mean_retraso_llegada=mean(atraso_llegada)), 
2     by=origen]
```

```
origen num_vuelos mean_retraso_llegada
<char>      <int>          <num>
1:   EWR      120835          NA
2:   LGA      104662          NA
3:   JFK      111279          NA
```



# Resumiendo por cada grupo

- Muchas variables pueden especificarse vía `.()` en `by` lo que significa usar cada combinación única para todas las variables como un grupo.

```
1 vuelos[, .(num_vuelos=.N, mean_retraso_llegada=mean(atraso_llegada)) ,
2     by=.(origen, destino)]
```

|      | origen | destino | num_vuelos | mean_retraso_llegada |
|------|--------|---------|------------|----------------------|
|      | <char> | <char>  | <int>      | <num>                |
| 1:   | EWR    | IAH     | 3973       | NA                   |
| 2:   | LGA    | IAH     | 2951       | NA                   |
| 3:   | JFK    | MIA     | 3314       | NA                   |
| 4:   | JFK    | BQN     | 599        | NA                   |
| 5:   | LGA    | ATL     | 10263      | NA                   |
| ---  |        |         |            |                      |
| 220: | LGA    | TVC     | 77         | NA                   |
| 221: | LGA    | MYR     | 3          | -3.0                 |
| 222: | EWR    | TVC     | 24         | NA                   |
| 223: | EWR    | ANC     | 8          | -2.5                 |
| 224: | EWR    | LGA     | 1          | NA                   |

# Nuevo by/agrupando variables

```

1 vuelos[, .(
2   num_vuelos=.N,
3   mean_retraso_llegada=mean(atraso_llegada)
4 ), by=.( 
5   origen=tolower(origen),
6   destino.punt=destino
7 )]

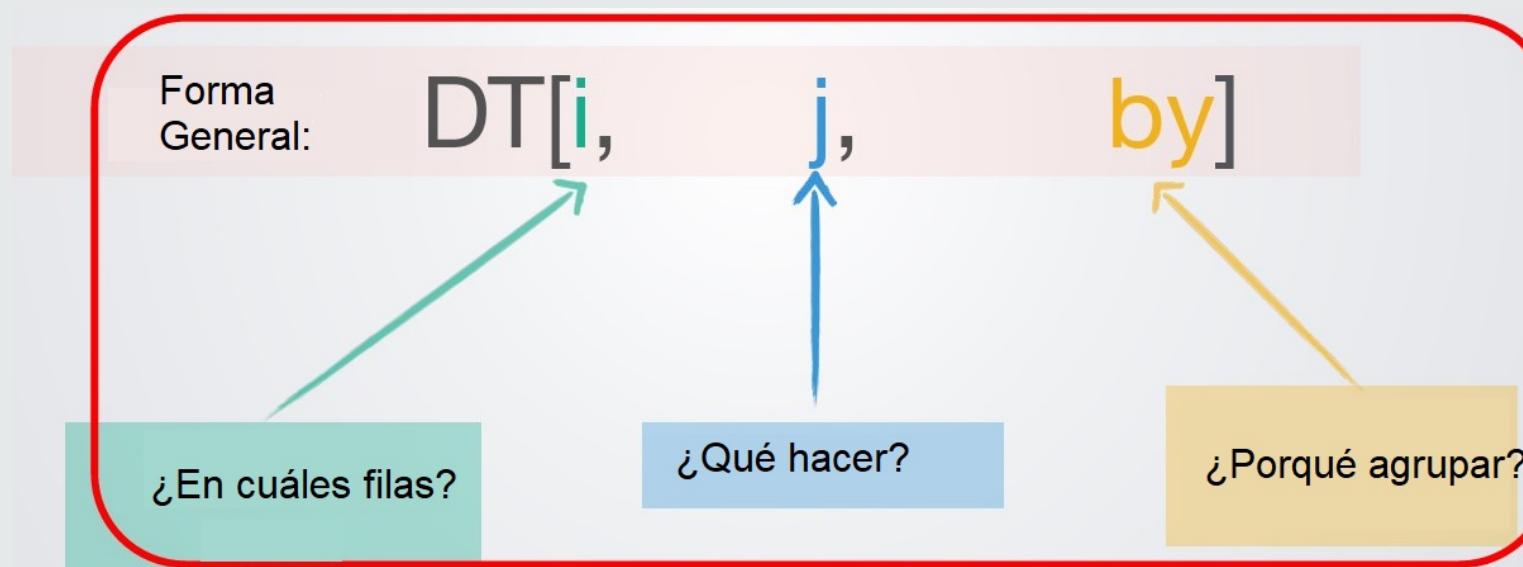
```

|      | origen | destino.punt | num_vuelos | mean_retraso_llegada |
|------|--------|--------------|------------|----------------------|
|      | <char> | <char>       | <int>      | <num>                |
| 1:   | ewr    | IAH          | 3973       | NA                   |
| 2:   | lga    | IAH          | 2951       | NA                   |
| 3:   | jfk    | MIA          | 3314       | NA                   |
| 4:   | jfk    | BQN          | 599        | NA                   |
| 5:   | lga    | ATL          | 10263      | NA                   |
| ---  |        |              |            |                      |
| 220: | lga    | TVC          | 77         | NA                   |
| 221: | lga    | MYR          | 3          | -3.0                 |
| 222: | ewr    | TVC          | 24         | NA                   |
| 223: | ewr    | ANC          | 8          | -2.5                 |
| 224: | ewr    | LGA          | 1          | NA                   |

- Las variables pueden renombrarse al computarse.
- Ejercicio: para todos los vuelos con retraso de salida positiva o negativa cual fue el promedio *mean* de las demoras de los arribos? Ayuda: usa sign(retraso\_ida) en by.

# Usando los tres argumentos juntos

- Piensa en términos básicos de unidades
  - filas
  - columnas
  - grupo
- La sintaxis de data.table provee un lugar para cada uno de ellos.



# Subconjunto de filas antes de resumir

- `i` puede ser usada en el mismo tiempo que `j` y `by`.
- Para el subconjunto de vuelos con AA como **aerolinea**, por cada origen y destino, obtén el número de vuelos, y el promedio *mean* de arribos demorados.

```
1 vuelos[aerolinea=="AA", .(num_vuelos=.N, mean_retraso_llegada=mean(atraso_llegada)), by=.(origen, destino)]
```

|     | origen | destino | num_vuelos | mean_retraso_llegada |
|-----|--------|---------|------------|----------------------|
|     | <char> | <char>  | <int>      | <num>                |
| 1:  | JFK    | MIA     | 2221       | NA                   |
| 2:  | LGA    | ORD     | 5694       | NA                   |
| 3:  | LGA    | DFW     | 4836       | NA                   |
| 4:  | EWR    | MIA     | 1068       | NA                   |
| 5:  | LGA    | MIA     | 3945       | NA                   |
| 6:  | JFK    | SJU     | 1099       | NA                   |
| 7:  | JFK    | MCO     | 730        | NA                   |
| 8:  | JFK    | FLL     | 182        | NA                   |
| 9:  | EWR    | DFW     | 2054       | NA                   |
| 10: | JFK    | LAX     | 3217       | NA                   |
| 11: | JFK    | SFO     | 1422       | NA                   |
| 12: | JFK    | BOS     | 1455       | NA                   |
| 13: | LGA    | STL     | 902        | NA                   |
| 14: | JFK    | STT     | 303        | NA                   |
| 15: | JFK    | DFW     | 367        | NA                   |
| 16: | JFK    | EGE     | 103        | NA                   |
| 17: | JFK    | LAS     | 639        | NA                   |
| 18: | JFK    | AUS     | 365        | NA                   |



# DT[i] dentro de j con by

- Para cada grupo, definido con un único valor de origen y destino, selecciona la primer fila, y las columnas tiempo\_vuelo, distancia.
- Ayuda: .() es sinónimo de list(), por lo que data.table() debe ser usado aquí.

```
1 vuelos[, data.table(tiempo_vuelo, distancia) [1] , by=.(origen,destino)]
```

|       | origen | destino | tiempo_vuelo | distancia |
|-------|--------|---------|--------------|-----------|
|       | <char> | <char>  | <num>        | <num>     |
| 1:    | EWR    | IAH     | 227          | 1400      |
| 2:    | LGA    | IAH     | 227          | 1416      |
| 3:    | JFK    | MIA     | 160          | 1089      |
| 4:    | JFK    | BQN     | 183          | 1576      |
| 5:    | LGA    | ATL     | 116          | 762       |
| <hr/> |        |         |              |           |
| 220:  | LGA    | TVC     | 101          | 655       |
| 221:  | LGA    | MYR     | 74           | 563       |
| 222:  | EWR    | TVC     | 93           | 644       |
| 223:  | EWR    | ANC     | 418          | 3370      |
| 224:  | EWR    | LGA     | NA           | 17        |

# SD[i] dentro de j con by

- Para cada grupo, definido de valores únicos de origen y destino, selecciona la primer fila, y todas las columnas.
- Ayuda: .SD es un data.table; lo que significa "[subconjunto de datos](#)" para un grupo.

```
1 vuelos[, .SD[1], by=.(origen,destino)]
```

|      | origen        | destino         | anio               | mes            | dia       | horario_salida | salida_programada |
|------|---------------|-----------------|--------------------|----------------|-----------|----------------|-------------------|
|      | <char>        | <char>          | <int>              | <int>          | <int>     | <int>          | <int>             |
| 1:   | EWR           | IAH             | 2013               | 1              | 1         | 517            | 515               |
| 2:   | LGA           | IAH             | 2013               | 1              | 1         | 533            | 529               |
| 3:   | JFK           | MIA             | 2013               | 1              | 1         | 542            | 540               |
| 4:   | JFK           | BQN             | 2013               | 1              | 1         | 544            | 545               |
| 5:   | LGA           | ATL             | 2013               | 1              | 1         | 554            | 600               |
| ---  |               |                 |                    |                |           |                |                   |
| 220: | LGA           | TVC             | 2013               | 6              | 14        | 1756           | 1800              |
| 221: | LGA           | MYR             | 2013               | 6              | 15        | 1517           | 1520              |
| 222: | EWR           | TVC             | 2013               | 7              | 5         | 1355           | 1400              |
| 223: | EWR           | ANC             | 2013               | 7              | 6         | 1629           | 1615              |
| 224: | EWR           | LGA             | 2013               | 7              | 27        | NA             | 106               |
|      |               |                 |                    |                |           |                |                   |
|      | atraso_salida | horario_llegada | llegada_programada | atraso_llegada | aerolinea |                |                   |
|      | <num>         | <int>           | <int>              | <num>          | <char>    |                |                   |
| 1:   | 2             | 830             | 819                | 11             | UA        |                |                   |
| 2:   | 4             | 850             | 830                | 20             | UA        |                |                   |
| 3:   | 2             | 923             | 850                | 33             | AA        |                |                   |
| 4:   | -1            | 1004            | 1022               | -18            | B6        |                |                   |
| 5:   | -6            | 812             | 837                | -25            | DL        |                |                   |
| ---  |               |                 |                    |                |           |                |                   |
| 220: | -4            | 2004            | 2005               | -1             | MQ        |                |                   |

# .SD[i] dentro de j con by

- Ejercicio: por cada origen y destino, obtene la **última fila**.

Ayuda: recordá que .N significa el número de filas en una tabla.

- Ejercicio: por cada origen y destino, obtené la **primer y última fila**.

# Escribir y leer CSV en pedazos

# fwrite - escribe eficiente CSV

-Escribe el data set completo de vuelos a un disco.

```
fwrite(vuelos, "vuelos.csv")
```

# DT[, j=fwrite(), by=variable]

- Escribe una fila por mes, 1.csv, 2.csv, etc. -by= mes significa que en j, mes es escalable, por lo que paste0 ( mes, “.csv”) puede ser usado para crear 1.csv, 2.csv, etc.

```
1 vuelos[, fwrite(
2   data.table(tiempo_vuelo, distancia),
3   paste0(mes, ".csv")
4 ), by=mes]
```

Empty data.table (0 rows and 1 cols): mes

```
1 Sys.glob("*csv")
[1] "1.csv"        "10.csv"       "11.csv"       "12.csv"
[5] "2.csv"        "3.csv"        "4.csv"        "5.csv"
[9] "6.csv"        "7.csv"        "8.csv"        "9.csv"
[13] "flights14.csv"
```

# fread CSV, combinado con rbindlist

- (1) inicialmente lista vacía, (2) en cada iteración del bucle for, asigna una tabla con un elemento de la lista, (3) rbindlist para crear una tabla simple con todos los datos.

```
1 mes.dt.list <- list()  
2  
3 # (2)  
4 for(mes.csv in Sys.glob("*.csv")){  
5   mes<- sub(".csv", "", mes.csv)  
6   mes.dt.list[[mes.csv]] <- data.table(mes,fread(mes.csv)) }  
7  
8 # (3)  (mes.dt <- data.table::rbindlist(mes.dt.list))
```



# Lee muchos CSV usando by

- by=mes.csv significa computar `j=fread(mes.csv)` por cada único valor de mes.csv (por cada archivo CSV)
- Los resultados están combinados vía `rbindlist` y se retornan como una tabla simple.

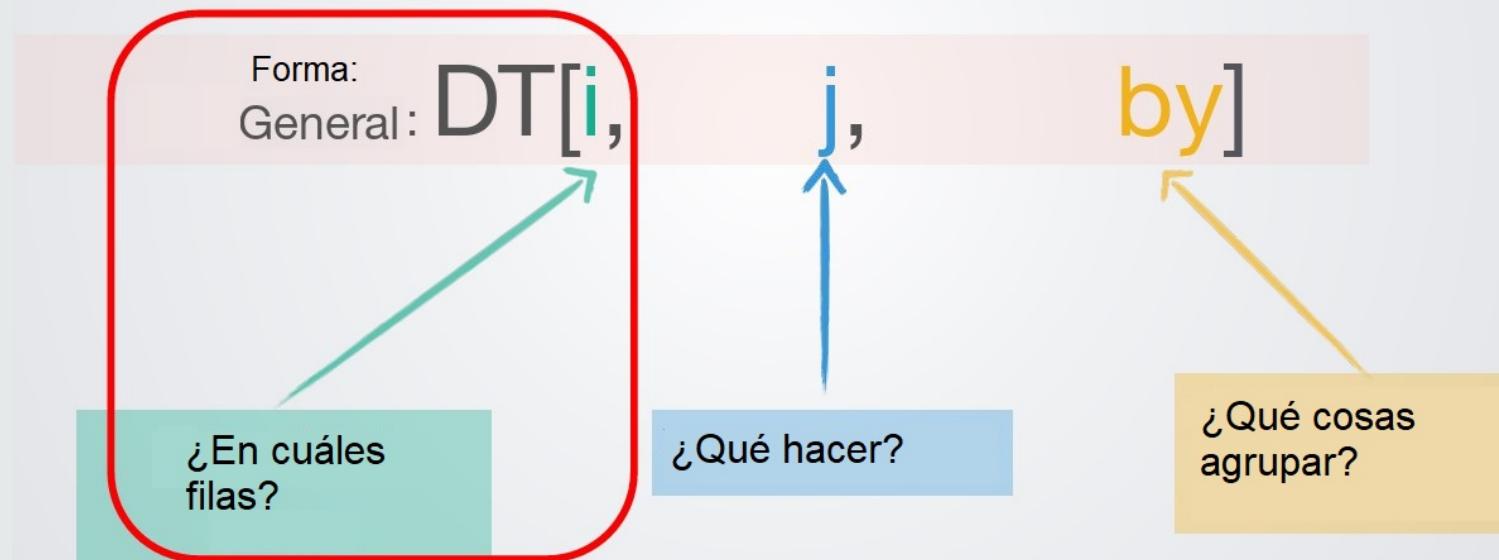
```
**data.table(mes.csv=Sys.glob("*.csv")) [, fread(mes.csv) , by=mes.csv]**
```

# Ejercicios

- Usa `vuelos[, fwrite(.SD), by=origen]` para escribir un archivo CSV para cada aeropuerto de Nueva York.
- Crea `CSV.dt`, a data.table de nombres de archivo CSV, usando `Sys.glob("*.csv")`
- Usa `CSV.dt[, fread(origen.csv), by=origen.csv]` para leerlos de vuelta a R.

# Uniendo: cuando i es un data.table

- Piensa en términos básicos de unidades
  - filas**
  - columnas**
  - grupo**
- La sintaxis de data.table provee un lugar único para cada uno de ellos.



# Une DT con la tabla i: DT[i]

- Retorna cada una de las filas en vuelos que coincide con una fila en tres\_dias,
- con una fila NA al final del día que no coincide.

```

1 tres_dias<- rbind(
2   data.table(mes=3, dia=17, evento="St.Patrick"),
3   data.table(mes=9, dia=26, evento="Cumpleaños"),
4   data.table(mes=12, dia=25, evento="Navidad")))

```

|    | mes   | dia   | evento     |
|----|-------|-------|------------|
|    | <num> | <num> | <char>     |
| 1: | 3     | 17    | St.Patrick |
| 2: | 9     | 26    | Cumpleaños |
| 3: | 12    | 25    | Navidad    |

```

1 vuelos[tres_dias, on=c("mes", "dia")]

```

|       | anio  | mes   | dia   | horario_salida  | salida_programada  | atraso_salida  |
|-------|-------|-------|-------|-----------------|--------------------|----------------|
|       | <int> | <int> | <int> | <int>           | <int>              | <num>          |
| 1:    | 2013  | 3     | 17    | 17              | 2253               | 84             |
| 2:    | 2013  | 3     | 17    | 27              | 2358               | 29             |
| 3:    | 2013  | 3     | 17    | 30              | 2355               | 35             |
| 4:    | 2013  | 3     | 17    | 31              | 2359               | 32             |
| 5:    | 2013  | 3     | 17    | 57              | 2100               | 237            |
| ---   |       |       |       |                 |                    |                |
| 2618: | 2013  | 12    | 25    | 2357            | 2359               | -2             |
| 2619: | 2013  | 12    | 25    | NA              | 1630               | NA             |
| 2620: | 2013  | 12    | 25    | NA              | 1525               | NA             |
| 2621: | 2013  | 12    | 25    | NA              | 1100               | NA             |
| 2622: | 2013  | 12    | 25    | NA              | 1930               | NA             |
|       |       |       |       | horario_llegada | llegada_programada | atraso_llegada |
|       |       |       |       | <int>           | <int>              | <num>          |
| 1:    |       |       |       | 134             | 14                 | 80             |
|       |       |       |       | aerolinea       | vuelo              | <char>         |
|       |       |       |       | B6              | 112                | <int>          |



|       |     |      |     |    |      |
|-------|-----|------|-----|----|------|
| 2:    | 355 | 338  | 17  | B6 | 707  |
| 3:    | 405 | 340  | 25  | B6 | 739  |
| 4:    | 403 | 338  | 25  | B6 | 727  |
| 5:    | 321 | 2327 | 234 | DL | 1247 |
| ---   |     |      |     |    |      |
| 2618: | 433 | 437  | -4  | B6 | 839  |

# Unir dos tablas, c/múltiples argumentos

- `mult="all"` está por defecto. Es decir, retorna todas las filas que coincidan.
- Al principio o al final hay otras opciones, retorna una fila por vuelo, para cada fila de los tres días.

```
1 vuelos[tres_dias, on=c("mes","dia"), mult="first"]

  anio   mes   dia horario_salida salida_programada atraso_salida
  <int> <int> <int>           <int>           <int>           <num>
1: 2013     3    17            17            2253            84
2: 2013     9    26            451            500            -9
3: 2013    12    25            456            500            -4
  horario_llegada llegada_programada atraso_llegada aerolinea vuelo
  <int>           <int>           <num>           <char> <int>
1:        134            14            80             B6   112
2:        619            648           -29             US  1877
3:        649            651            -2             US  1895
  codigo_cola origen destino tiempo_vuelo distancia hora minuto
  <char> <char>  <char>       <num>      <num> <num> <num>
1: N583JB    JFK    BUF       56       301    22    53
2: N155UW    EWR    CLT       72       529     5    0
3: N156UW    EWR    CLT       98       529     5    0
  fecha_hora evento
  <POSc>   <char>
1: 2013-03-17 22:00:00 St.Patrick
2: 2013-09-26 05:00:00 Cumpleaños
3: 2013-12-25 05:00:00 Navidad
```



```
1 vuelos[tres_dias, on=c("mes","dia"), mult="last"]
```

|    | anio            | mes                | dia            | horario_salida | salida_programada | atraso_salida |        |
|----|-----------------|--------------------|----------------|----------------|-------------------|---------------|--------|
|    | <int>           | <int>              | <int>          | <int>          | <int>             | <num>         |        |
| 1: | 2013            | 3                  | 17             | NA             | 650               | NA            |        |
| 2: | 2013            | 9                  | 26             | NA             | 1240              | NA            |        |
| 3: | 2013            | 12                 | 25             | NA             | 1930              | NA            |        |
|    | horario_llegada | llegada_programada | atraso_llegada | aerolinea      | vuelo             |               |        |
|    | <int>           | <int>              | <num>          | <char>         | <int>             |               |        |
| 1: | NA              | 827                | NA             | UA             | 883               |               |        |
| 2: | NA              | 1525               | NA             | WN             | 4720              |               |        |
| 3: | NA              | 2115               | NA             | MQ             | 3535              |               |        |
|    | codigo_cola     | origen             | destino        | tiempo_vuelo   | distancia         | hora          | minuto |
|    | <char>          | <char>             | <char>         | <num>          | <num>             | <num>         | <num>  |
| 1: | <NA>            | EWR                | ORD            | NA             | 719               | 6             | 50     |
| 2: | N691WN          | EWR                | HOU            | NA             | 1411              | 12            | 40     |
| 3: | N509MQ          | JFK                | CMH            | NA             | 483               | 19            | 30     |
|    | fecha_hora      | evento             |                |                |                   |               |        |
|    | <POSc>          | <char>             |                |                |                   |               |        |
| 1: | 2013-03-17      | 06:00:00           | St.Patrick     |                |                   |               |        |
| 2: | 2013-09-26      | 12:00:00           | Cumpleaños     |                |                   |               |        |
| 3: | 2013-12-25      | 19:00:00           | Navidad        |                |                   |               |        |

# Uniendo dos tablas, sin filas NA

- `nomatch=0L` para retornar a 0 filas cuando no hay coincidencias. (en vez de 1 NA fila)

```
1 vuelos[tres_dias, on=c("mes","dia"), nomatch=0L]

  anio mes dia horario_salida salida_programada atraso_salida
  <int> <int> <int>           <int>           <int>       <num>
1: 2013   3   17             17            2253        84
2: 2013   3   17             27            2358        29
3: 2013   3   17             30            2355        35
4: 2013   3   17             31            2359        32
5: 2013   3   17             57            2100        237
---
2618: 2013  12   25           2357           2359        -2
2619: 2013  12   25             NA           1630        NA
2620: 2013  12   25             NA           1525        NA
2621: 2013  12   25             NA           1100        NA
2622: 2013  12   25             NA           1930        NA
  horario_llegada llegada_programada atraso_llegada aerolinea vuelo
  <int>           <int>       <num>     <char> <int>
1:      134            14          80      B6    112
2:      355            338          17      B6    707
3:      405            340          25      B6    739
4:      403            338          25      B6    727
5:      321            2327         234      DL  1247
---
2618:      433            437          -4      B6    839
```



# Uniendo dos tablas usando setkey

- `setkey(DT, col1, col2)` usa para ordenar DT por col1 y col2
- Las uniones son rápidas, y no necesitan que se especifiquen con `on=`, si los datos ya tienen `key`

```

1 setkey(tres_dias, mes, dia)
1 setkey(vuelos, mes, dia)
1 vuelos[tres_dias]

Key: <mes, dia>
      anio   mes   dia horario_salida salida_programada atraso_salida
      <int> <int> <int>           <int>           <int>           <num>
1: 2013     3    17          17            2253            84
2: 2013     3    17          27            2358            29
3: 2013     3    17          30            2355            35
4: 2013     3    17          31            2359            32
5: 2013     3    17          57            2100            237
---
2618: 2013    12    25          2357            2359            -2
2619: 2013    12    25            NA            1630            NA
2620: 2013    12    25            NA            1525            NA
2621: 2013    12    25            NA            1100            NA
2622: 2013    12    25            NA            1930            NA
horario_llegada llegada_programada atraso_llegada aerolinea vuelo
      <int>           <int>           <num>       <char> <int>

```



# Uniendo dos tablas de otra forma

- Uniendo de otra manera siempre devuelve al menos una fila por cada vuelo, NA si no coincide.
- Recuerda: `DT[i]` significa que retorna una fila por cada elemento en `i`.

```
1  (join.dt <- tres_dias[vuelos, on=c("mes","dia")][order(is.na(evento))])
```

|         | mes           | dia             | evento             | anio           | horario_salida | salida_programada |
|---------|---------------|-----------------|--------------------|----------------|----------------|-------------------|
|         | <int>         | <int>           | <char>             | <int>          | <int>          | <int>             |
| 1:      | 3             | 17              | St.Patrick         | 2013           | 17             | 2253              |
| 2:      | 3             | 17              | St.Patrick         | 2013           | 27             | 2358              |
| 3:      | 3             | 17              | St.Patrick         | 2013           | 30             | 2355              |
| 4:      | 3             | 17              | St.Patrick         | 2013           | 31             | 2359              |
| 5:      | 3             | 17              | St.Patrick         | 2013           | 57             | 2100              |
| ---     |               |                 |                    |                |                |                   |
| 336772: | 12            | 31              | <NA>               | 2013           | NA             | 705               |
| 336773: | 12            | 31              | <NA>               | 2013           | NA             | 825               |
| 336774: | 12            | 31              | <NA>               | 2013           | NA             | 1615              |
| 336775: | 12            | 31              | <NA>               | 2013           | NA             | 600               |
| 336776: | 12            | 31              | <NA>               | 2013           | NA             | 830               |
|         |               |                 |                    |                |                |                   |
|         | atraso_salida | horario_llegada | llegada_programada | atraso_llegada |                |                   |
|         | <num>         | <int>           | <int>              | <num>          |                |                   |
| 1:      | 84            | 134             | 14                 | 80             |                |                   |
| 2:      | 29            | 355             | 338                | 17             |                |                   |
| 3:      | 35            | 405             | 340                | 25             |                |                   |
| 4:      | 32            | 403             | 338                | 25             |                |                   |
| 5:      | 237           | 321             | 2327               | 234            |                |                   |



# Ejercicios

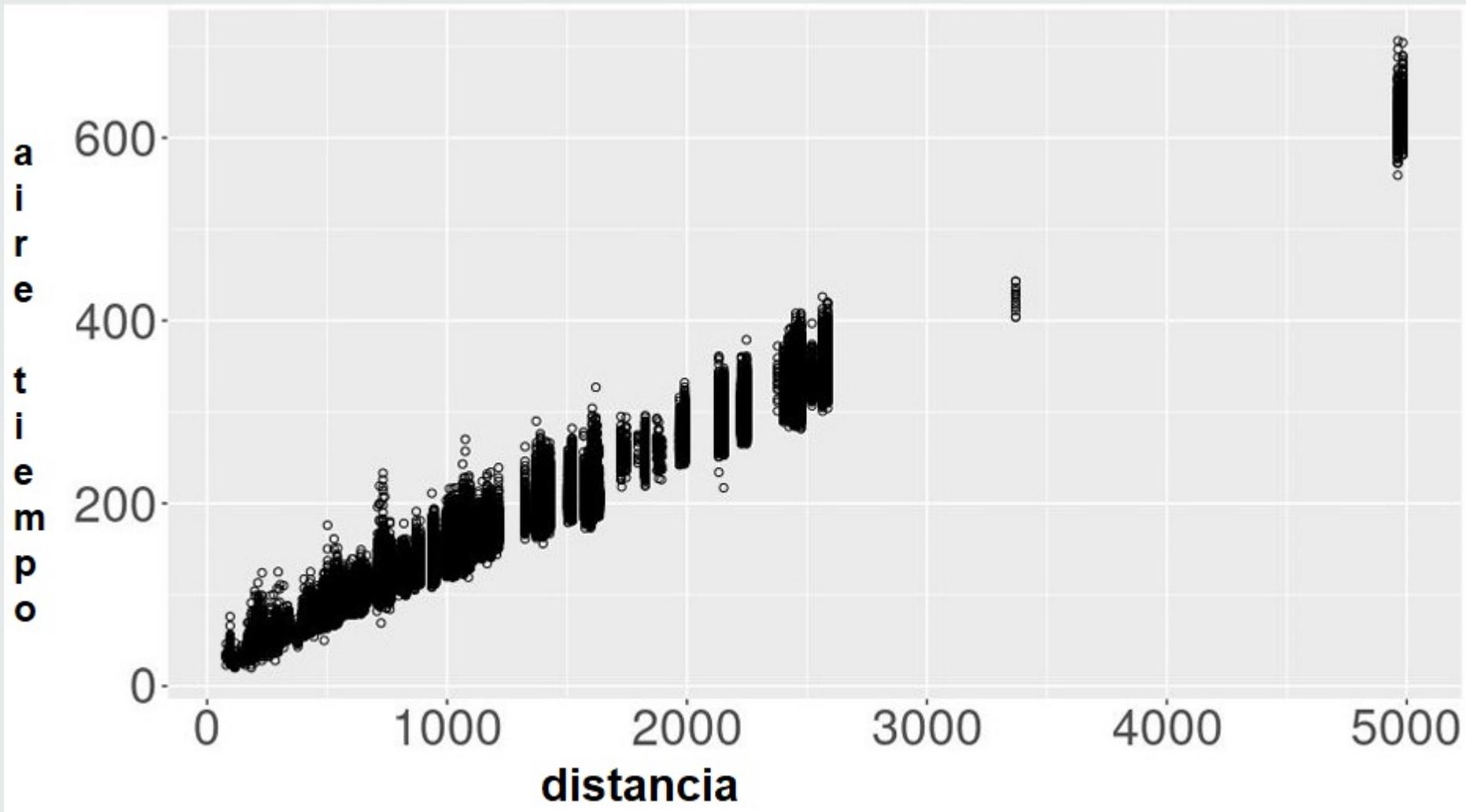
. Crea una tabla de datos de aeropuertos de la siguiente forma:

|   | codigo | nombre    |
|---|--------|-----------|
| 1 | JFK    | kennedy   |
| 2 | LGA    | Laguardia |
| 3 | EWR    | Newark    |

- Une usando vuelos[airports, `on=.` (`origen=codigo`) ] que significa unir para unirse a la columna de origen en vuelos con el código columnas en aeropuertos.

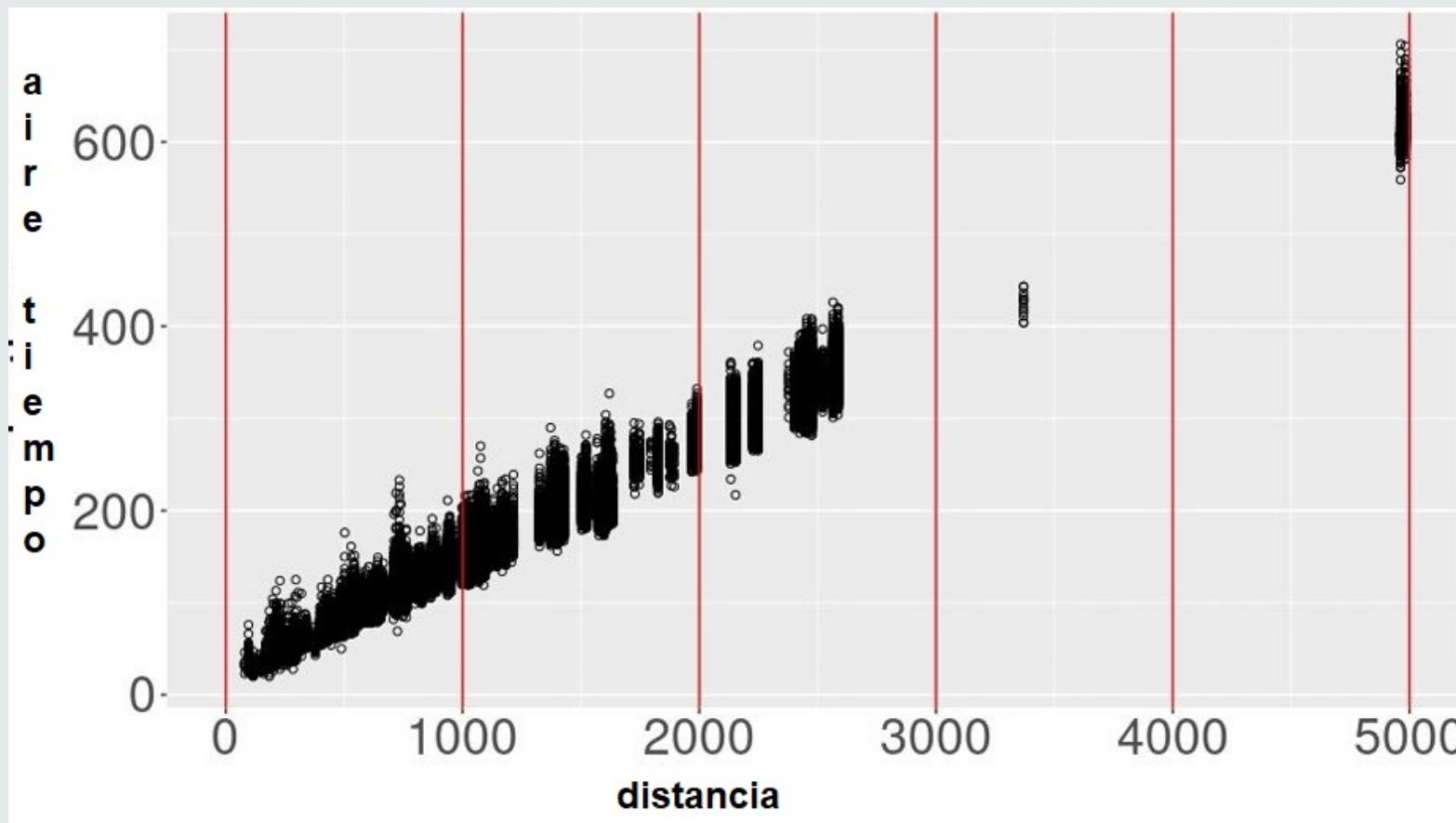
# Cómo computar un histograma de las distancias

- o mean por cada distancia (0,1000, 2000, etc)



# Define los contenedores del histograma

```
1 grid.point <- seq(0, 5000, by=1000)
2 grid.dt <- data.table(grid.point, distancia=grid.point)
```



# Rolling join, roll=Inf

- Cada punto de la cuadrícula coincide con los vuelos que ocurren después.

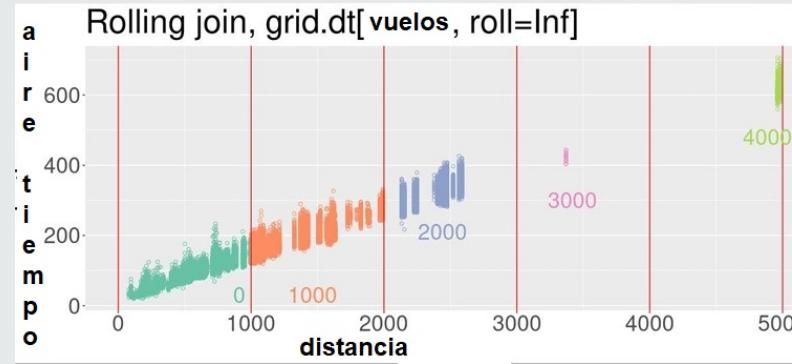
```

1 setkey(grid.dt, distancia)
2 setkey(vuelos, distancia)
3 (join.dt <- grid.dt[vuelos, roll=Inf])

```

Key: <distancia>

|         | grid.point     | distancia         | anio  | mes               |
|---------|----------------|-------------------|-------|-------------------|
| dia     | horario_salida | salida_programada | <num> | <num> <int> <int> |
| <int>   | <int>          | <int>             |       |                   |
| 1:      | 0              | 17                | 2013  | 7                 |
| 27      | NA             |                   | 106   |                   |
| 2:      | 0              | 80                | 2013  | 1                 |
| 3       | 2127           |                   | 2129  |                   |
| 3:      | 0              | 80                | 2013  | 1                 |
| 4       | 1240           |                   | 1200  |                   |
| 4:      | 0              | 80                | 2013  | 1                 |
| 4       | 1829           |                   | 1615  |                   |
| 5:      | 0              | 80                | 2013  | 1                 |
| 4       | 2128           |                   | 2129  |                   |
| ---     |                |                   |       |                   |
| 336772: | 4000           | 4983              | 2013  | 12                |
| 27      | 922            |                   | 930   |                   |
| 336773: | 4000           | 4983              | 2013  | 12                |
| 28      | 933            |                   | 930   |                   |
| 336774: | 4000           | 4983              | 2013  | 12                |
| 29      | 920            |                   | 930   |                   |



# Rolling join, roll=-Inf

- Cada punto de la cuadrícula coincide con los vuelos que ocurren antes.

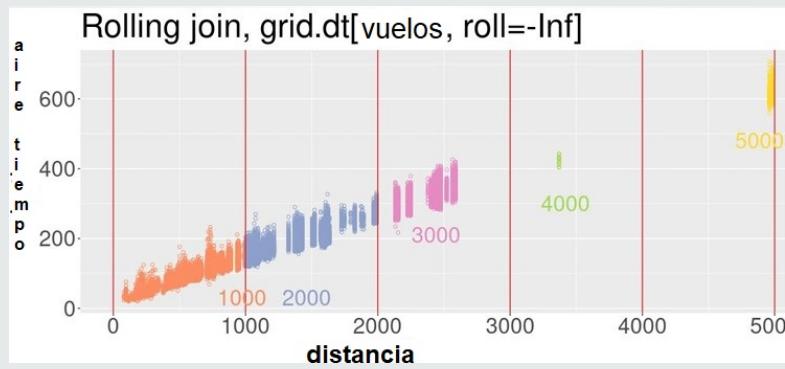
```

1  setkey(grid.dt, distancia)
2  setkey(vuelos, distancia)
3  (join.dt <- grid.dt[vuelos, roll=-Inf])

```

Key: <distancia>

|         | grid.point     | distancia         | anio  | mes               |
|---------|----------------|-------------------|-------|-------------------|
| dia     | horario_salida | salida_programada | <num> | <num> <int> <int> |
|         | <int>          | <int>             |       | <int>             |
| 1:      | 1000           | 17                | 2013  | 7                 |
| 27      | NA             |                   | 106   |                   |
| 2:      | 1000           | 80                | 2013  | 1                 |
| 3       | 2127           |                   | 2129  |                   |
| 3:      | 1000           | 80                | 2013  | 1                 |
| 4       | 1240           |                   | 1200  |                   |
| 4:      | 1000           | 80                | 2013  | 1                 |
| 4       | 1829           |                   | 1615  |                   |
| 5:      | 1000           | 80                | 2013  | 1                 |
| 4       | 2128           |                   | 2129  |                   |
| ---     |                |                   |       |                   |
| 336772: | 5000           | 4983              | 2013  | 12                |
| 27      | 922            |                   | 930   |                   |
| 336773: | 5000           | 4983              | 2013  | 12                |
| 28      | 933            |                   | 930   |                   |
| 336774: | 5000           | 4983              | 2013  | 12                |
| 29      | 920            |                   | 930   |                   |



# Rolling join, roll="nearest"

- Cada punto de la cuadrícula coincide con los vuelos que ocurren más cerca.

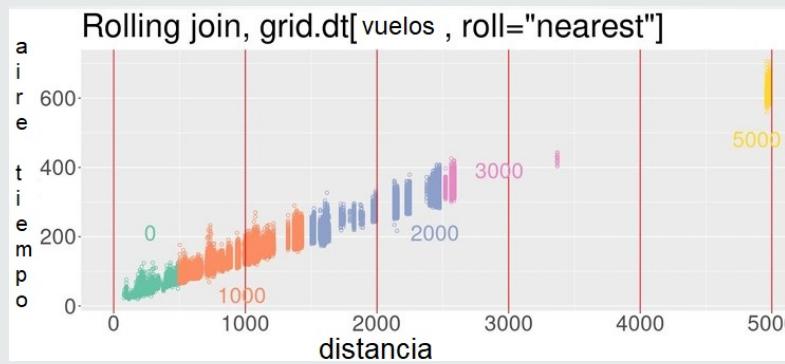
```

1 setkey(grid.dt, distancia)
2 setkey(vuelos, distancia)
3 (join.dt <- grid.dt[vuelos, roll="nearest"])

```

Key: <distancia>

|         | grid.point     | distancia         | anio  | mes   |
|---------|----------------|-------------------|-------|-------|
| dia     | horario_salida | salida_programada | <num> | <num> |
|         | <int>          | <int>             | <int> | <int> |
| 1:      | 0              | 17                | 2013  | 7     |
| 27      | NA             |                   | 106   |       |
| 2:      | 0              | 80                | 2013  | 1     |
| 3       | 2127           |                   | 2129  |       |
| 3:      | 0              | 80                | 2013  | 1     |
| 4       | 1240           |                   | 1200  |       |
| 4:      | 0              | 80                | 2013  | 1     |
| 4       | 1829           |                   | 1615  |       |
| 5:      | 0              | 80                | 2013  | 1     |
| 4       | 2128           |                   | 2129  |       |
| ---     |                |                   |       |       |
| 336772: | 5000           | 4983              | 2013  | 12    |
| 27      | 922            |                   | 930   |       |
| 336773: | 5000           | 4983              | 2013  | 12    |
| 28      | 933            |                   | 930   |       |
| 336774: | 5000           | 4983              | 2013  | 12    |
| 29      | 920            |                   | 930   |       |

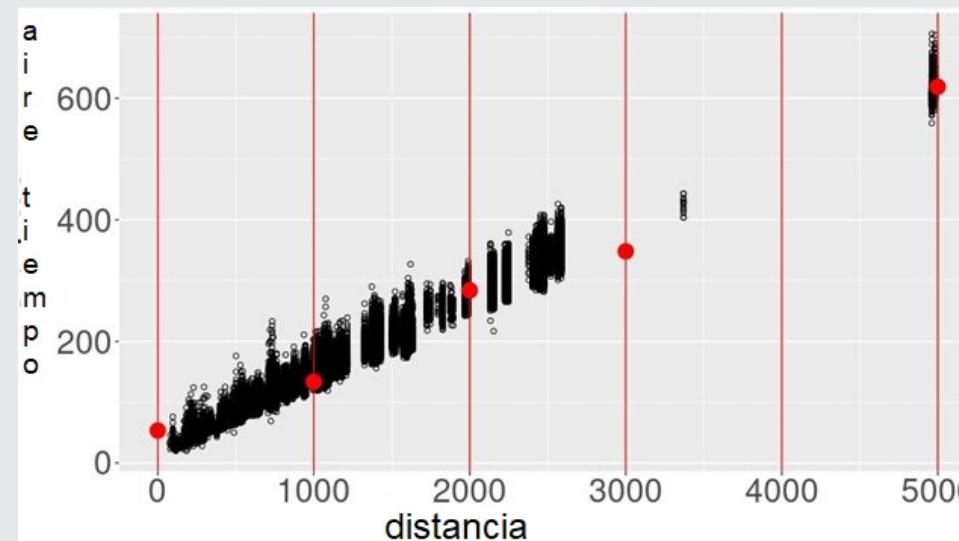


# Resumiendo después de: rolling join

- ¿Cómo computar un resumen/histograma?

```
1 join.dt[, .(num_vuelos=.N, mean_minutes=mean(tiempo_vuelo)), by=grid.point]
```

|    | grid.point | num_vuelos | mean_minutes |
|----|------------|------------|--------------|
|    | <num>      | <int>      | <num>        |
| 1: | 0          | 80327      | NA           |
| 2: | 1000       | 183736     | NA           |
| 3: | 2000       | 57742      | NA           |
| 4: | 3000       | 14264      | NA           |
| 5: | 5000       | 707        | NA           |



# No-igualdades une con on=.

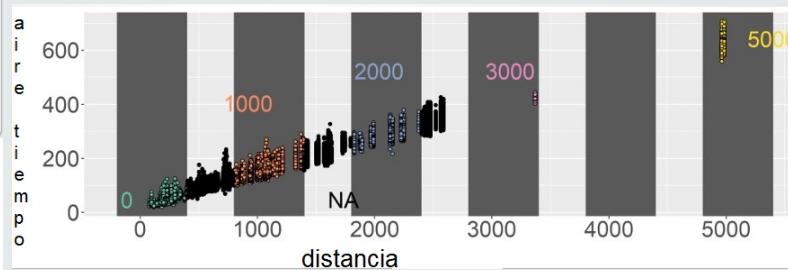
## (DTcol < icol)

```
1 rect.dt <- data.table(
2   grid.point,
3   min_dist=grid.point-200,
4   max_dist=grid.point+400))
```

|    | grid.point | min_dist | max_dist |
|----|------------|----------|----------|
|    | <num>      | <num>    | <num>    |
| 1: | 0          | -200     | 400      |
| 2: | 1000       | 800      | 1400     |
| 3: | 2000       | 1800     | 2400     |
| 4: | 3000       | 2800     | 3400     |
| 5: | 4000       | 3800     | 4400     |
| 6: | 5000       | 4800     | 5400     |

```
1 head(join.dt <- rect.dt[vuelos, .(
2   distancia, tiempo_vuelo, grid.point
3 ), on=.(  
4   min_dist<distancia, max_dist>distanci
```

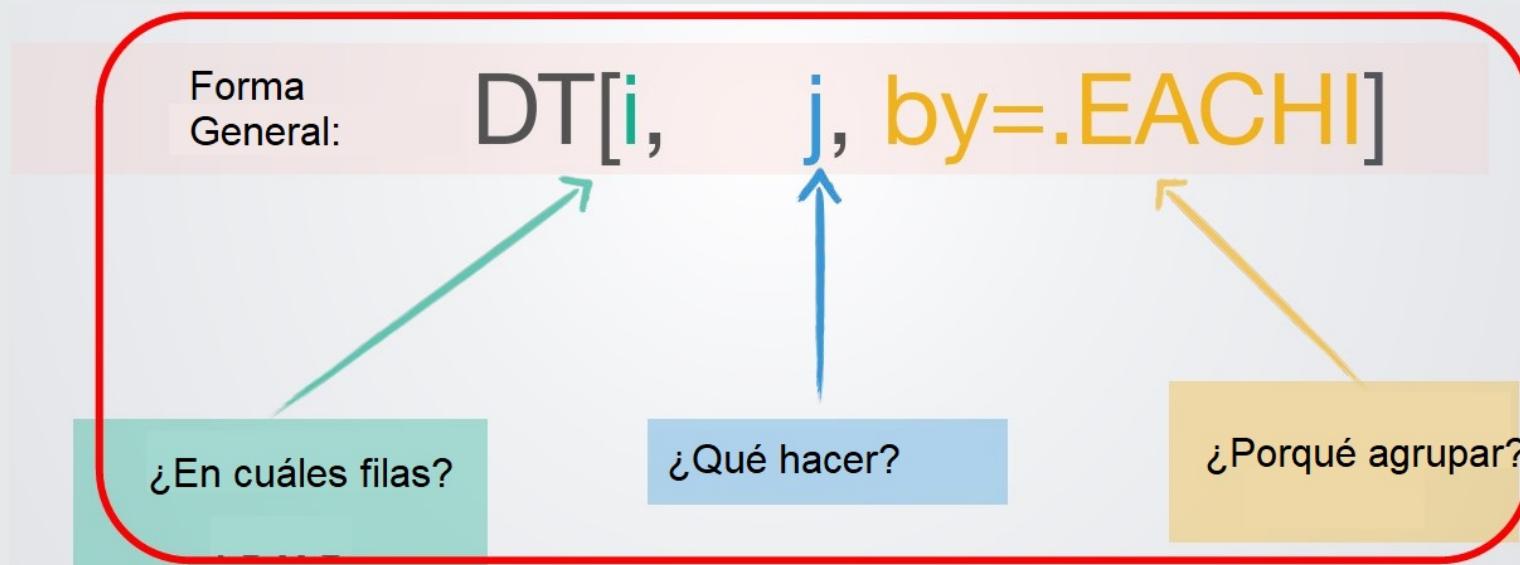
|    | distancia | tiempo_vuelo | grid.point |
|----|-----------|--------------|------------|
|    | <num>     | <num>        | <num>      |
| 1: | 17        | NA           | 0          |
| 2: | 80        | 30           | 0          |
| 3: | 80        | 30           | 0          |
| 4: | 80        | 28           | 0          |
| 5: | 80        | 32           | 0          |
| 6: | 80        | 29           | 0          |



Une todas las filas que satisfacen las desigualdades dadas.

# Computa por cada fila en la tabla i

- Piensa en términos básicos de unidades  
**filas**  
**columnas**  
**grupo**
- La sintaxis de data.table provee un lugar único para cada uno de ellos.



# Resumir uniendo via by=.EACHI

```

1 vuelos[tres_dias, .(
2   mean_aire_time=mean(tiempo_vuelo),
3   num_vuelos=N
4 ), by=.EACHI, on=.(mes,dia)]

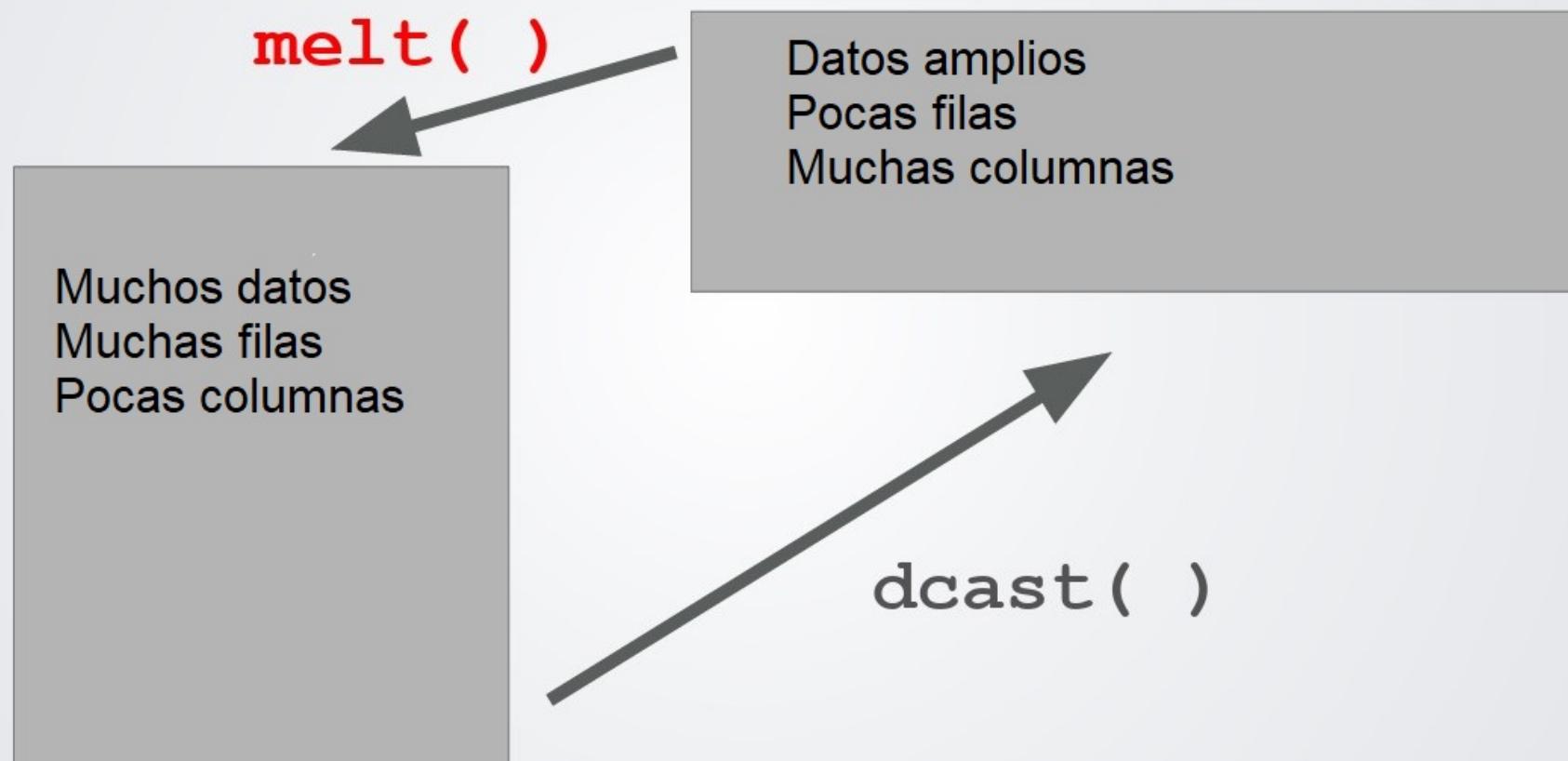
```

Key: <mes, dia>

|    | mes   | dia   | mean_aire_time | num_vuelos |
|----|-------|-------|----------------|------------|
|    | <int> | <int> | <num>          | <int>      |
| 1: | 3     | 17    | NA             | 907        |
| 2: | 9     | 26    | NA             | 996        |
| 3: | 12    | 25    | NA             | 719        |

- `DT[ i, j, by=.EACHI, on]` significa que por cada fila en i, miramos las filas en DT la cual coinciden usando en columnas, y computar/retornando j.
- No es lo mismo que `by=(mes,día)` la cual hace el cómputo por cada combinación única de mes y día (mucho menos eficiente si solo quisiéramos el resultado para unas pocas combinaciones)

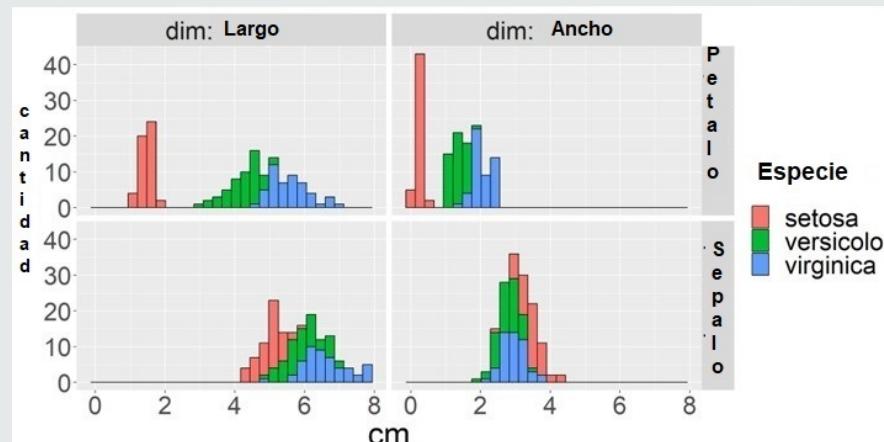
# Remodelación de datos: ancho a largo (melt)



Similar a `stats::reshape(dirección="long")`, `tidyr::pivot_longer()`

# melt: remodelación de datos de ancho a largo

- ¿Cómo hacer el gráfico de abajo con datos de flores?
- La tabla de datos flores tiene columnas Largo.Sepalo, Ancho.Sepalo, Largo.Petalo, Ancho.Petalo, Especie.
- Necesitaríamos `facet_grid(part ~ dim) + geom_histogram(aes(cm, fill=Especie))`,
- Donde la columna dim tiene valores Largo o Ancho, parte de los valores son Petalo o Sepalo



# melt: remodelación de ancho a largo de los datos

```
melt(DT,id.vars =c("especie","flores"))
```

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Convertir 4 entradas de medidas para 1 columna de salida y 1 columna variable de salida

# melt: remodelación de ancho a largo de los datos

```
melt(DT,measure.vars=c("Largo.Sepalo","Ancho.Sepalo", "Largo.Petalo", "Ancho.Petalo))
```

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Convertir 4 entradas de medidas para 1 columna de salida y 1 columna variable de salida



# melt: remodelación de ancho a largo de los datos

```
melt(DT,measure.vars=patterns(".[.]"))
```

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Convertir 4 entradas de medidas para 1 columna de salida y 1 columna variable de salida



# melt: remodelación en largos datos

```
melt(DT, id.vars=c("ID1","ID2"))
```

```
melt(DT, measure.vars=c("valueA","valueB"))
```

```
melt(DT, measure.vars=patterns("regular expression"))
```

- Ejercicio: `melt` las columnas `atraso_llegada` y `atraso_salida` de los datos de vuelos, especificando las medidas como `measure.vars`
- Adicionalmente especifica `id.vars` que limita con las columnas y hace copia de la salida `output`.



# melt con measure( ), nuevo en GitHub

`**melt(DT, measure.vars=measure(part,dim, pattern="(.*)."))**`

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Convertir 4 entradas de medidas para 1 columna de salida y 1 columna variable de salida



# melt con measure( ), nuevo en GitHub

```
**melt(DT, measure.vars=measure(part,dim, pattern=(.*).“))**
```

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

# melt con measure(), nuevo en GitHub

```
melt(DT,measure.vars=measure(part, dim, sep=".")
```

Usos de columna de entradas con mas grupos después de reachicar(2 en este caso)

Tipo de col.Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

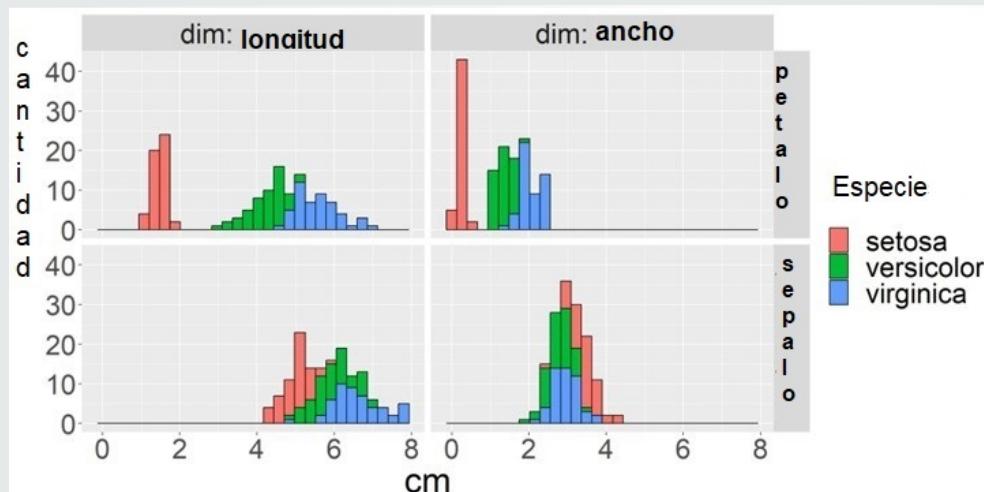
|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Convertir 4 entradas de medidas para 1 columna de salida y 1 columna variable de salida



# melt con measure(), nuevo en GitHub

- Ejercicio: hace un gráfico con los datos de flores.
- Primero convierte flores en un data.table, luego reacomoda la forma usando `melt(DT, measure=measure(-----), value.name="cm")`



```
library(ggplot2); ggplot() + facet_grid(part ~ dim) + geom_histogram(aes(cm, fill=Especie), data=output_of_melt)
```

# melt con measure(), nueva en GitHub

```
melt(DT, measure.vars=measure(part, value.name, sep=":"))
```

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Salida del valor de la columna por cada dim, y columna variable por parte.

# melt con measure(), nuevo en GitHub

```
melt(DT, measure.vars=measure(value.name, dim, sep=":"))
```

Tipo de col. Leyenda: |Nombre| = medida/valor, |Nombre| = variable, |Nombre| = id

|    | Largo.Sepalo | Ancho.Sepalo | Largo.Petalo | Ancho.Petalo | Especie flores |
|----|--------------|--------------|--------------|--------------|----------------|
| 1: | 5.1          | 3.5          | 1.4          | 0.2          | setosa 1       |
| 2: | 7.0          | 3.2          | 4.7          | 1.4          | versicolor 51  |

Amplia entrada con 4 columnas de medidas

Largo con 1 columna de valor

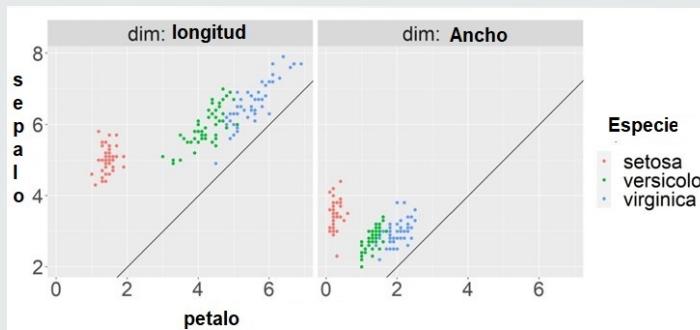
|    | valor | especie | flores | variable     |
|----|-------|---------|--------|--------------|
| 1: | 5.1   | setosa  | 1      | Largo.Sepalo |
| 2: | 3.5   | setosa  | 1      | Ancho.Sepalo |

Salida del valor de la columna por cada part, y columna variable por dim.



# melt con medida (), nueva en GitHub

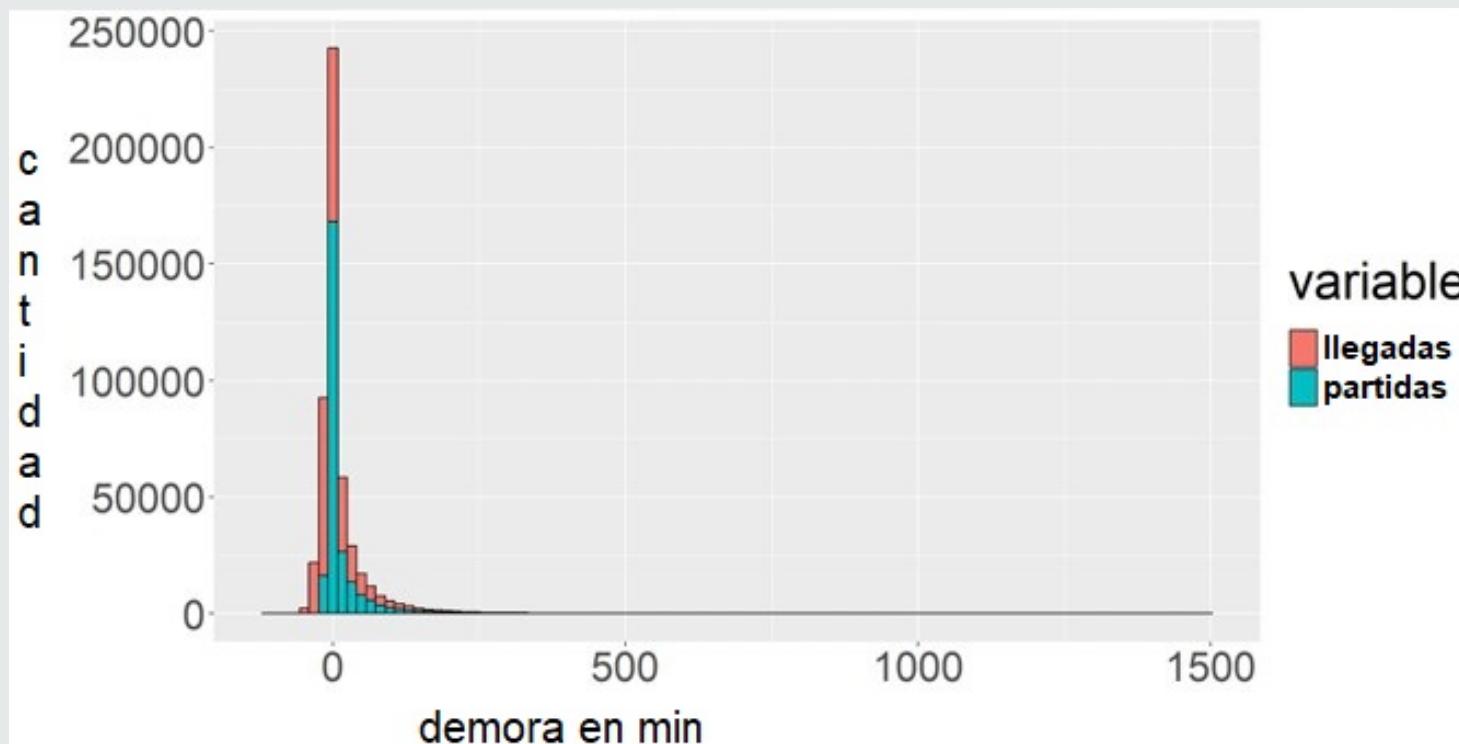
- Ejercicio: para mostrar que sepalo es más larga que petalo, hace un gráfico con los datos flores. Primero convertí flores en data.table.
- Luego usa melt con measure() para reajustar.
- `library(ggplot2); ggplot() + facet_grid(.~dim) + geom_point(aes( Petalo, Sepalo, color=Especie), data=output_of_melt)+ geom_abline( slope=1, intercept=0)`



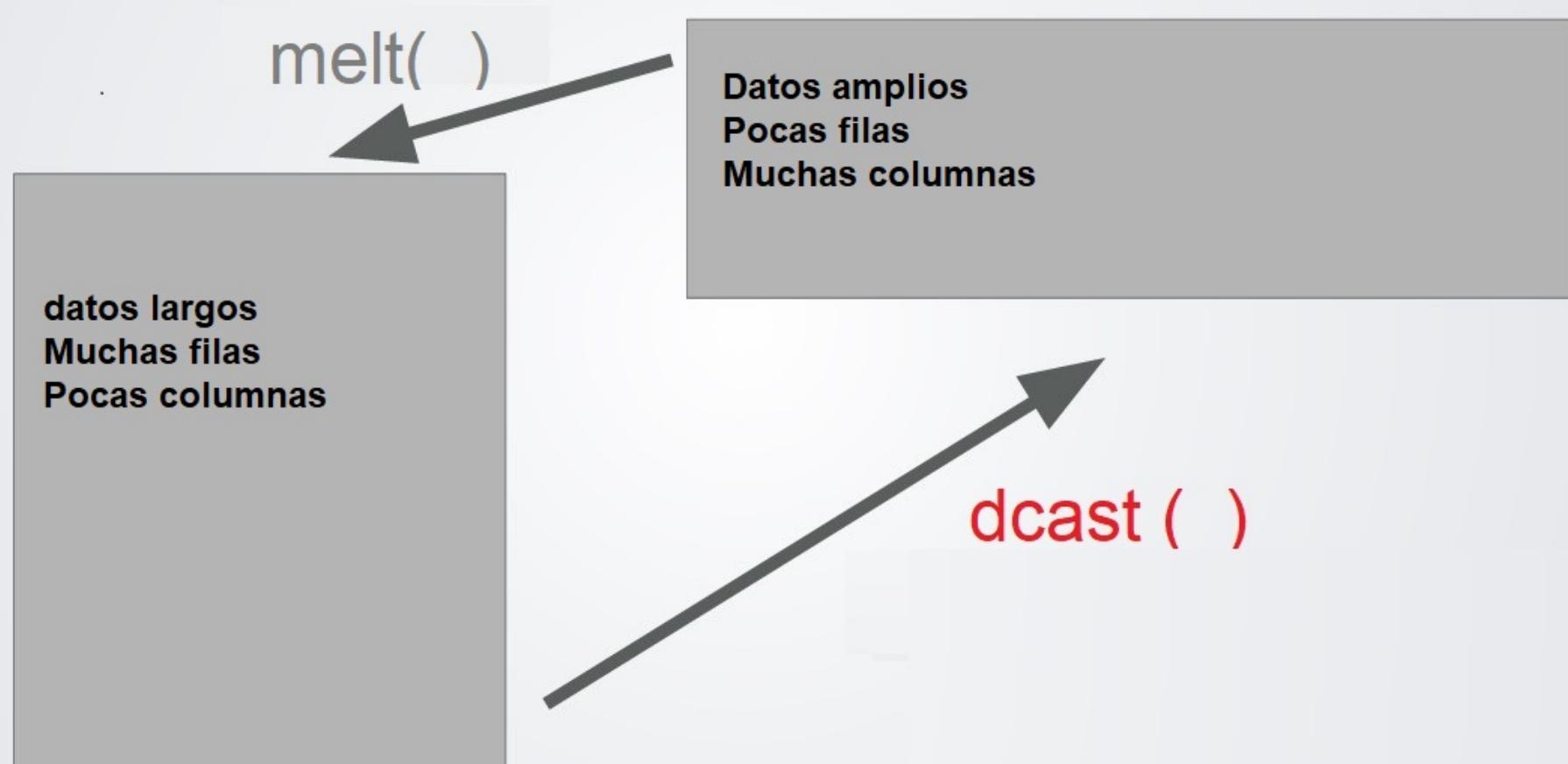
Ejemplos de fusión adaptados de: Hocking TD. Datos de ancho a alto remodelar usando expresiones regulares y el nc paquete. (2021), doi:10.32614/RJ-2021-029

# melt ejercicios para los datos de vuelos

- Ejercicio: hace un gráfico abajo con los datos de vuelos.
- Usa melt con ambos patterns() o measure() para readaptar la forma.
- ggplot() + geom\_histogram(aes(demora\_minutos, fill=variable), data=output\_of\_melt)



# Reajustando datos: larga forma (dcast)



similar a `stats::reshape(direction="wide")`, `tidyr::pivot_wider()`

# dcast: reajuste de largas formas

- **dcast** puede obtener la versión original del datos de flores volviendo de la versión grande.

```
1 flores.dt=data.table(flores) [, flores := .I] # es decir número de fila en data.table
```

```
1 flores.largo=melt(flores.dt,measure=measure(part,dim,sep="."))
2 head(flores.largo)
```

|    | Especie | flores | part   | dim    | value |
|----|---------|--------|--------|--------|-------|
|    | <fctr>  | <int>  | <char> | <char> | <num> |
| 1: | setosa  | 1      | Largo  | Sepalo | 5.1   |
| 2: | setosa  | 2      | Largo  | Sepalo | 4.9   |
| 3: | setosa  | 3      | Largo  | Sepalo | 4.7   |
| 4: | setosa  | 4      | Largo  | Sepalo | 4.6   |
| 5: | setosa  | 5      | Largo  | Sepalo | 5.0   |
| 6: | setosa  | 6      | Largo  | Sepalo | 5.4   |

```
1 # La formula indica donde esta puesto los diferentes valores de la variable: rows #~ cols
2
3 flores.ancho=dcast(flores.largo, flores ~ part + dim, sep=".")
4 head(flores.ancho)
```

Key: <flores>

|    | flores | Ancho.Petalo | Ancho.Sepalo | Largo.Petalo | Largo.Sepalo |
|----|--------|--------------|--------------|--------------|--------------|
|    | <int>  | <num>        | <num>        | <num>        | <num>        |
| 1: | 1      | 0.2          | 3.5          | 1.4          | 5.1          |
| 2: | 2      | 0.2          | 3.0          | 1.4          | 4.9          |
| 3: | 3      | 0.2          | 3.2          | 1.3          | 4.7          |



# dcast: reajuste de largas formas

- dcast puede computarse a resumenes/funciones agregadas

```
1 flores.largo=melt(flores.dt,measure=measure(part,dim,sep=".") ,value.name="cm")
2 flores.largo
```

|      | Especie   | flores | part   | dim    | cm    |
|------|-----------|--------|--------|--------|-------|
|      | <fctr>    | <int>  | <char> | <char> | <num> |
| 1:   | setosa    | 1      | Largo  | Sepalo | 5.1   |
| 2:   | setosa    | 2      | Largo  | Sepalo | 4.9   |
| 3:   | setosa    | 3      | Largo  | Sepalo | 4.7   |
| 4:   | setosa    | 4      | Largo  | Sepalo | 4.6   |
| 5:   | setosa    | 5      | Largo  | Sepalo | 5.0   |
| ---  |           |        |        |        |       |
| 596: | virginica | 146    | Ancho  | Petalo | 2.3   |
| 597: | virginica | 147    | Ancho  | Petalo | 1.9   |
| 598: | virginica | 148    | Ancho  | Petalo | 2.0   |
| 599: | virginica | 149    | Ancho  | Petalo | 2.3   |
| 600: | virginica | 150    | Ancho  | Petalo | 1.8   |

```
1 dcast(flores.largo, Especie ~ dim, fun.aggregate=mean, value.var="cm")
```

|    | Key: <Especie> | Especie | Petalo | Sepalo |
|----|----------------|---------|--------|--------|
|    |                | <fctr>  | <num>  | <num>  |
| 1: | setosa         | 0.854   | 4.217  |        |
| 2: | versicolor     | 2.793   | 4.353  |        |
| 3: | virginica      | 3.789   | 4.781  |        |



# dcast: reajustes de largas formas

- dcast puede computar muchos resúmenes/funciones agregadas

```
1 flores.largo=melt(flores.dt,measure=measure(part,dim,sep=".") ,value.name="cm")
2 flores.largo
```

|      | Especie   | flores | part   | dim    | cm    |
|------|-----------|--------|--------|--------|-------|
|      | <fctr>    | <int>  | <char> | <char> | <num> |
| 1:   | setosa    | 1      | Largo  | Sepalo | 5.1   |
| 2:   | setosa    | 2      | Largo  | Sepalo | 4.9   |
| 3:   | setosa    | 3      | Largo  | Sepalo | 4.7   |
| 4:   | setosa    | 4      | Largo  | Sepalo | 4.6   |
| 5:   | setosa    | 5      | Largo  | Sepalo | 5.0   |
| ---  |           |        |        |        |       |
| 596: | virginica | 146    | Ancho  | Petalo | 2.3   |
| 597: | virginica | 147    | Ancho  | Petalo | 1.9   |
| 598: | virginica | 148    | Ancho  | Petalo | 2.0   |
| 599: | virginica | 149    | Ancho  | Petalo | 2.3   |
| 600: | virginica | 150    | Ancho  | Petalo | 1.8   |

```
1 dcast(flores.largo, Especie ~ dim, fun.aggregate=list(min, mean, max)) #value.var="cm")
```

|    | Key: <Especie> | Especie       | cm_min_Petalo | cm_min_Sepalo | cm_mean_Petalo | cm_mean_Sepalo |
|----|----------------|---------------|---------------|---------------|----------------|----------------|
|    |                | <fctr>        | <num>         | <num>         | <num>          | <num>          |
| 1: | setosa         | setosa        | 0.1           | 2.3           | 0.854          | 4.217          |
| 2: | versicolor     | versicolor    | 1.0           | 2.0           | 2.793          | 4.353          |
| 3: | virginica      | virginica     | 1.4           | 2.2           | 3.789          | 4.781          |
|    |                | cm_max_Petalo | cm_max_Sepalo |               |                |                |
|    |                | <num>         | <num>         |               |                |                |



|    |     |     |
|----|-----|-----|
| 1: | 1.9 | 5.8 |
| 2: | 5.1 | 7.0 |
| 3: | 6.9 | 7.9 |

# dcast: reajuste de largas formas

- dcast puede computar muchos valores de columnas.

```
1 flores.largo=melt(flores.dt,measure=measure(part,value.name,sep="."))
2 flores.largo
```

|      | Especie   | flores | part   | Sepalo | Petalo |
|------|-----------|--------|--------|--------|--------|
|      | <fctr>    | <int>  | <char> | <num>  | <num>  |
| 1:   | setosa    | 1      | Largo  | 5.1    | 1.4    |
| 2:   | setosa    | 2      | Largo  | 4.9    | 1.4    |
| 3:   | setosa    | 3      | Largo  | 4.7    | 1.3    |
| 4:   | setosa    | 4      | Largo  | 4.6    | 1.5    |
| 5:   | setosa    | 5      | Largo  | 5.0    | 1.4    |
| ---  |           |        |        |        |        |
| 296: | virginica | 146    | Ancho  | 3.0    | 2.3    |
| 297: | virginica | 147    | Ancho  | 2.5    | 1.9    |
| 298: | virginica | 148    | Ancho  | 3.0    | 2.0    |
| 299: | virginica | 149    | Ancho  | 3.4    | 2.3    |
| 300: | virginica | 150    | Ancho  | 3.0    | 1.8    |

```
1 dcast(flores.largo, Especie + part ~ ., fun.aggregate=list(min, max), value.var=c("Sepalo", "Petalo"))
```

|    | Especie    | part   | Sepalo_min | Petalo_min | Sepalo_max | Petalo_max |
|----|------------|--------|------------|------------|------------|------------|
|    | <fctr>     | <char> | <num>      | <num>      | <num>      | <num>      |
| 1: | setosa     | Ancho  | 2.3        | 0.1        | 4.4        | 0.6        |
| 2: | setosa     | Largo  | 4.3        | 1.0        | 5.8        | 1.9        |
| 3: | versicolor | Ancho  | 2.0        | 1.0        | 3.4        | 1.8        |
| 4: | versicolor | Largo  | 4.9        | 3.0        | 7.0        | 5.1        |



# Ejercicios

|   | destino | EWR | JFK | LGA |
|---|---------|-----|-----|-----|
| 1 | ABQ     | 0   | 278 | 0   |
| 2 | ACK     | 0   | 277 | 0   |
| 3 | AGS     | 0   | 0   | 0   |
| 4 | ALB     | 169 | 0   | 0   |
| 5 | ANC     | 13  | 0   | 0   |

- Hace una table como la de arriba, con una columna por cada origen, y una fila por cada destino(cuenta las entradas de los vuelos). Ayuda: usa destino~origen para crear un destino diferente en cada fila, y un diferente origen en cada columna
- Especifica value.var=c(“atraso\_salida”,“atraso\_llegada”) con fun.aggregate=mean para comparar el promedio de retrados. Usa esta información para elegir el mejor aeropuerto por cada destino.

# Importa data.table en tu paquete

- data.table actualmente es importada para 1400+ paquetes en CRAN.  
-“Imports:data.table” en la descripción.
- “import(data.table)” en NAMESPACE.
- .<- j\_variable <- NULL en la primer linea en la función de R para abordarCRAN. NOTA acerca de una función/variable no visible.
- data.table::setDTthreads(2) en ejemplos/testeos para abordar CRAN NOTA: poco tiempo necesita (por defecto data.table usa 1/2 de todo el CPUs)

# NOTA 1 CRAN

-`<-variable<-NULL` en la primer linea de la función de R para abordar  
CRAN NOTA: acerca de funciones/variable no visible.

```
1 mi_funcion <- function(DT) DT[, .(m=mean(x)) , by=y]
```

\*\* Chequea el código de R por posibles problemas... algunas notas para que consideres:

mi\_funcion: si no está visible como función global por `.`

mi\_funcion: si no está visible como función global `variablex`

mi\_funcion: si no está visible como función global variable `y`

- Chequea la nota como arriba, usa el código R de aquí abajo:

```
1 mi_funcion <- function(DT) {  
2   x <- y <- . <- NULL  
3   DT[, .(m=mean(x)), by=y]  
4 }
```

# NOTA2: CRAN

- CRAN requiere de paquetes para usar max 2CPUs durante chequeos.
- data.table por defecto usa 1/2 de todos los CPUs de la máquina.

```
Flavor: r-devel-linux-x86_64-debian-gcc
Check: examples, Result: NOTE
Examples with CPU time > 2.5 times elapsed time
              user  system elapsed ratio
aum_line_search      12.349   0.322   1.935 6.548
aum_line_search_grid 10.033   0.308   1.781 5.806
aum_diffs_penalty     4.730   0.169   1.635 2.996
```

- Chequea la NOTA como arriba, usa el código R como en los ejemplos de arriba y testeos:

```
1 data.table:::setDTthreads(2)
```

4/4 Contribuyendo con  
data.table

# Comunidad blog y encuesta

- La mascota de data.table es un león marino que ladra “R R R”
- La comunidad data.table tiene un nuevo blog, The Raft,

<https://rdatatable-community.github.io/The-Raft/>

Los leones marinos a menudo flotan juntos en la superficie del océano en grupos llamadas(balsas) ”rafts.” - Centro de Mamíferos Marinos.

- Por favor completa la encuesta.

<https://tinyurl.com/datatable-survey>

~10 minutos, cuentanos acerca del uso en data.table!  
(ayudanos a entender qué priorizar en el futuro)



# Repositorio GitHub

- data.table tiene un activo *issues/Pull Request(PR)* de extracción (PR)  
<https://github.com/Rdatatable/data.table/>
- 1000+ open issues, 100+ open PRs
- si tu tienes algo de tiempo/interés,podríamos usar tu ayuda! -Es fácil contribuir: prueba reproducir en problemas (De mucha ayuda para saber si hay algún problemas que sean reproducible)
- Es una comunidad muy inclusiva luego de tu ingreso por primera vez en PR, serás invitado a unirte al grupo de github!
- Ahora se vienen tiempos muy exitantes para involucrarte,ya que actualmente estamos creando un formal Documento escrito que describe la gobernanza descentralizada del proyecto, #5676



# Premios de Traducción

- En 2023-2025, la Fundación Nacional de Ciencias ha proporcionado fondos para apoyar y ampliar el ecosistema de usuarios y contribuyentes en torno a data.table
- 20 premios de traducción, de US\$ 500 cada uno, para realizar documentación y mensajes más accesibles, ideas:
- Traducir errores/advertencias/mensajes (el paquete potools puede ayudar)
- Traducir las viñetas más importantes (introducción, importación, remodelación)
- Traducir otra documentación (hojas de referencia, diapositivas, etc.)
- Prioridad: portugués, español, chino, francés, ruso, árabe, hindi
  - Llamado para propuestas: <https://rdatatable-community.github.io/The-Raft/>



# Premios en Viajes

- En 2023-2025, la Fundación Nacional de Ciencias ha proporcionado fondos para apoyar y ampliar el ecosistema de usuarios y contribuyentes en torno a data.table
- Ocho premios de viaje, 2700 dólares cada uno.
- Los candidatos deben dar una charla sobre data.table en una conferencia con una audiencia relevante (posibles usuarios o contribuyentes de data.table)
- Convocatoria de propuestas próximamente <https://rdatatable-community.github.io/The-Raft/>

# Resumen de data.table

- Sintaxis concisa y consistente.
- Rápida , eficiente en memoria.
- No tiene dependencias ( es fácil de instalar)
- No hay quiebres en los cambios ( es fácil de actualizar)
- Inclusive el usuario/desarrollador de la comunidad tiene oportunidades de contribuir:
- Premios en traducción, US\$500 cada uno.
- Premios en viajes, US\$2700 cada uno.

# GRACIAS. ¿PREGUNTAS?

Mara Destefanis

maragdestefanis@gmail.com



Fundado por el programa NSF POSE,  
proyecto #2303612. diapositivas adaptadas  
de Toby Dylan Hocking, Arun Srinivasan, y  
datatable-viñeta de introducción - gracias!



