

Programa de Doctorado

Tidyverse



Manejo y visualización de datos en R

Paloma Ruiz Benito, Verónica Cruz Alonso,
Julen Astigarraga Urcelay

Enero - 2022

Contenido

Parte I.- Introducción a R

- Entornos de programación, introducción a R y RStudio.
- Tipos de variables y datos, operaciones aritméticas y lógicas, creación de vectores, matrices, listas y tablas. Selección de datos.

Parte II.- Manejo de datos

- Flujo y funciones en la gestión de bases de datos.
- Recomendaciones para la estructura de las bases de datos y creación de código de programación.
- Introducción a la gestión de datos. Recomendaciones para su generación.
- Estructuras de programación: condicionales, bucles y funciones.

Parte III.- Visualización de datos

- Funciones básicas e introducción a ggplot para la visualización de datos.
- Generación de gráficos unidimensionales: histograma, dispersión, gráfico de cajas y bigotes, etc.
- Generación de gráficos bidimensionales: dispersión, boxplot, gráficos de barras, etc.
- Ejemplos y prácticas de visualización de gráficos en mapas.

Parte IV.- Trabajo reproducible

- Trabajo reproducible.
- Introducción a git y github.
- Rmarkdown.

Día 2



R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

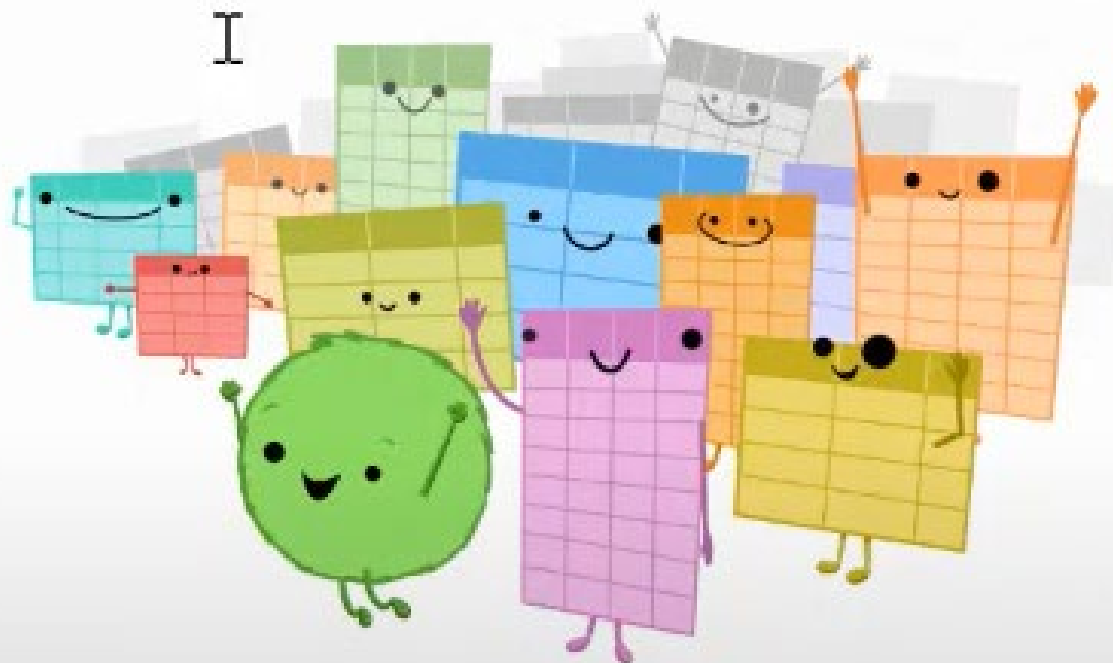
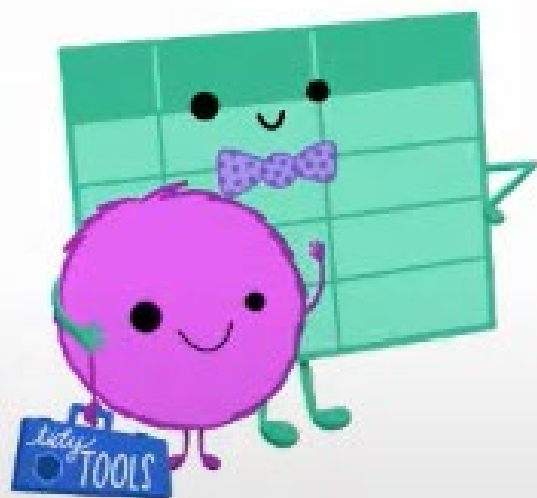
```
install.packages("tidyverse")
```

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

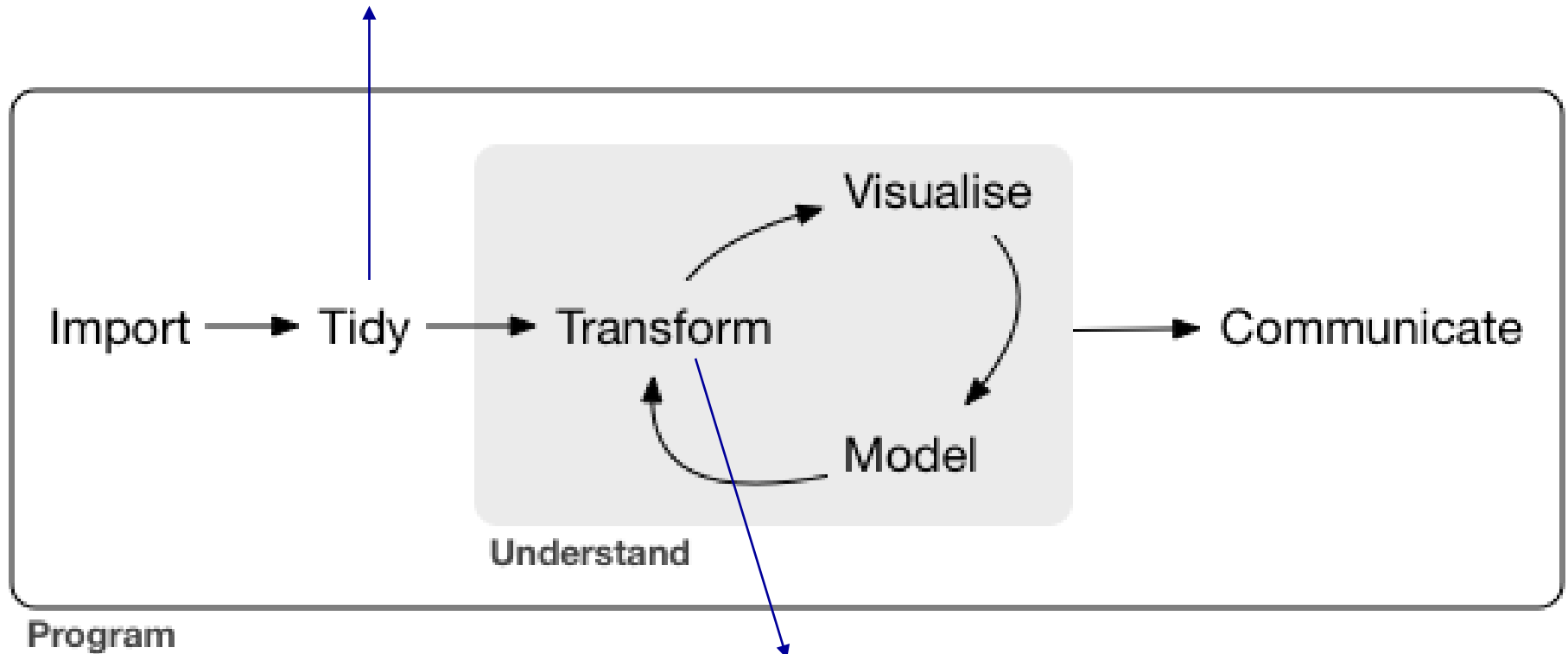
<https://www.kaggle.com/jessemostipak/dive-into-dplyr-tutorial-1>

https://github.com/Julenasti/intro_tidyverse-dplyr

I



Tidy data: cada columna es una variable y cada fila es una observación.



Transform: reducir el número de observaciones de interés, nuevas variables, calcular estadísticas

“**TIDY DATA** is a standard way of mapping the meaning of a dataset to its structure.”

—HADLEY WICKHAM

In tidy data:


- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable



id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

each row an observation





Todos los paquetes tienen una misma estructura
(filosofía, gramática y estructura de datos)



Recommended book: [R for Data Science](<https://r4ds.had.co.nz/>)
Hadley Wickham & Garrett Grolemund

+ **Core tidyverse packages:**

- + **[readr]**(<https://readr.tidyverse.org/>): lectura de datos
- + **[dplyr]**(<https://dplyr.tidyverse.org/>): manipulación de datos
- + **[tibble]**(<https://tibble.tidyverse.org/>): uso moderno del data.frame
- + **[tidyr]**(<https://tidyr.tidyverse.org/>): ordenando los datos
- + **[purrr]**(<https://purrr.tidyverse.org/>): programación funcional
- + **[stringr]**(<https://stringr.tidyverse.org/>): manipulación de cadenas
- + **[forcats]**(<https://forcats.tidyverse.org/>): trabajando con factores
- + **[ggplot2]**(<https://ggplot2.tidyverse.org/>): visualización de datos

La pipa %>%



La pipa %>%

¿De dónde viene?

¿Por qué se hizo así?



La pipa %>%

1. Punto de vista matemático
2. Punto de vista otros lenguajes de programación
3. Punto de vista del lenguaje de R

La pipa %>%

1. Punto de vista matemático

Pasas el resultado intermedio a la siguiente función

$$(f \circ g)(x) = f(g(x))$$

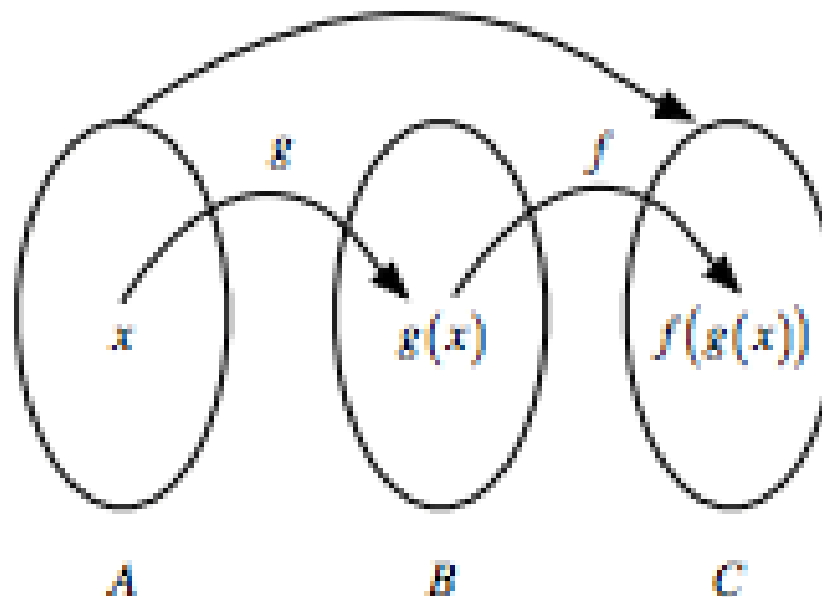


Image Credit: James Balamuta, ["Piping Data"](#)

La pipa %>%

2. Punto de vista otros lenguajes de programación

Shell or Terminal: pipeline character | o

La pipa %>%

3. Punto de vista del lenguaje de R

How can you implement F#'s forward pipe operator in R? The operator makes it possible to easily chain a sequence of calculations. For example, when you have an input data and want to call functions foo and bar in sequence, you can write:
`data > foo > bar`

<http://adolfoalvarez.cl/plumbers-chains-and-famous-painters-the-history-of-the-pipe-operator-in-r/>

La pipa %>%



{magrit} introdujo la pipa %>% en 2014

R 4.1.0 (2021) ha introducido a native pipe operator |>
(similar pero más eficiente)

La pipa %>%

**¿Por qué debemos
usarla?**



La pipa %>%

```
library(tidyverse)
```

```
#base
```

```
tapply(mtcars$mpg, mtcars$cyl, mean)
```

```
tapply(mtcars$mpg, mtcars$cyl, sd)
```

```
#tidyverse
```

```
result <- group_by(mtcars, cyl)
```

```
result <- summarise(result, meanMPG = mean(mpg))
```

```
#tidyverse & pipe
```

```
result <- mtcars %>%
```

```
  group_by(cyl) %>%
```

```
  summarise(meanMPG = mean(mpg),
```

```
            sdMPG = mean(mpg))
```

La pipa %>%

```
library(tidyverse)
```

```
head(summary(as.factor(rownames(mtcars))),2)
```

```
mtcars %>%  
  rownames %>%  
  as.factor() %>%  
  summary() %>%  
  head(n = 2)
```

La pipa %>%

Es más fácil de leer y ejecutar. Si usamos “%>%” como **entonces**, el código previo es muy fácil de comprender y las instrucciones sencillas.

- Evitamos repetición de código
- Evitamos tener que leer de dentro a fuera o de izquierda a derecha
- Podemos encadenar múltiples pasos

La pipa %>%

CTRL + SHIFT + M

use **@R!**