Ajuntament de Barcelona

# Data Massage

## Introduction to Data Massage

IT Academy

Barcelona Activa

Unió Europea
Fons Europeu de Desenvolupament Regional

# Index

- Introduction
- What does it mean for your data to be “tidy”?
- Common tasks
- Potential activity transforming your data
- Best practices

# Introduction

**It is rare that you get the data in exactly the right form you need**

What if the database is not formatted in the way you expect? Or the data is completely unstructured?

- **Before data is loaded to visualize it**, **it must be transformed** to meet any format and structural requirements.

- ***Data massaging*, also known as data *cleansing* or *scrubbing*, is a process that eliminates unnecessary information from data or cleans a dataset to make it useable.**

# Introduction

- **Databases come in different shapes and sizes and each must be treated as unique.**
- A few data massaging techniques are required to adapt the data to the algorithms we are working with.
- Common tasks include stripping unwanted characters and whitespace, converting number and date values into desired formats, and organising data into a meaningful structure.
- **Massaging the data is usually the "transform" step. In most cases, one or more transformations are required.**

# What does it mean for your data to be “tidy”?

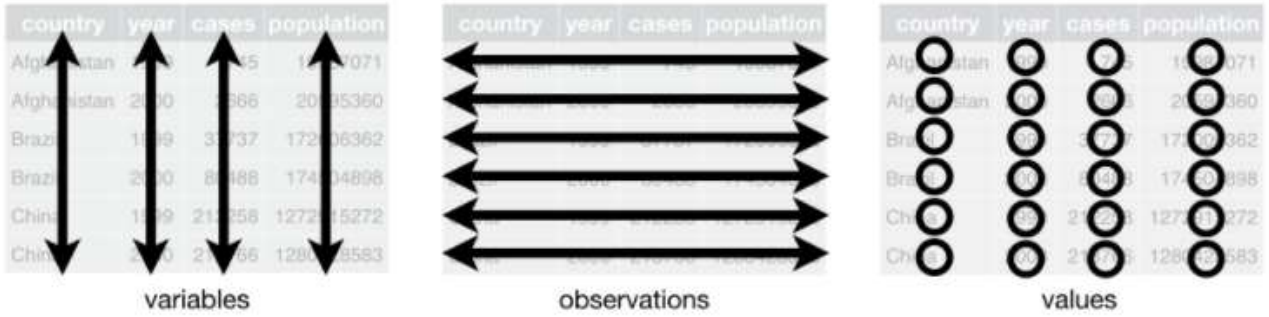
**The word “tidy” in data science using R means that your data follows a standardized format:**

- A dataset is a collection of values, usually either numbers (if quantitative) or strings AKA text data (if qualitative/categorical). **Values are organised in two ways. Every value belongs to a variable and an observation.**
- **A variable contains all values that measure the same underlying attribute across units** (examples: weight, temperature, duration).
- **An observation contains all values measured on the same unit across attributes** (examples: person, day, village).
- **“Tidy” data is a standard way of mapping the meaning of a dataset to its structure.** A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types.

# What does it mean for your data to be “tidy”?

**In “tidy data”:**

- Each variable forms a column
- Each observation forms a row
- Each type of observational unit forms a table




Tidy data graphic from R for Data Science

# What does it mean for your data to be “tidy”?

**Example:**

| Fecha | Nombre | Mate | Ingles |
|---|---|---|---|
| 1-11-2015 | Hernandez, Rodrigo | 90 | 60 |

| mes | año | primer | apellido | materia | puntos |
|---|---|---|---|---|---|
| 11 | 2015 | Rodrigo | Hernandez | mate | 90 |
| 11 | 2015 | Rodrigo | Hernandez | ingles | 60 |

Edgar Ruiz 2018

On the right table:

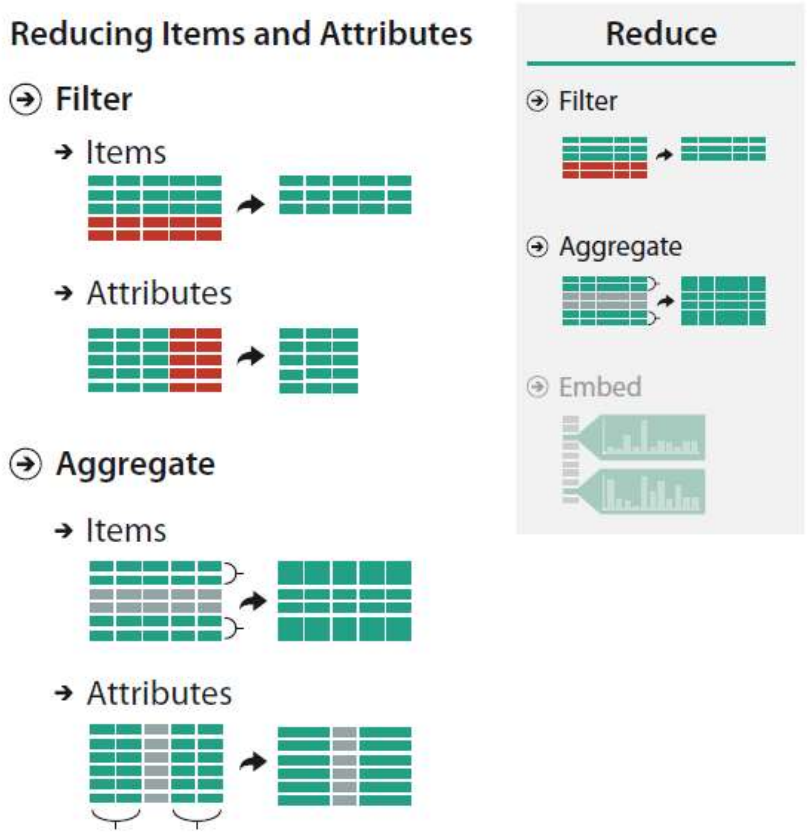
- Each variable forms a column
- Each observation forms a row

# Common tasks

**Things we do to massage the data include:**

- **Change formats** from the standard source system emissions to the target system requirements, e.g. change date format from m/d/y to d/m/y, or sort the data.
- **Replace missing values** with defaults, e.g. "0" when a quantity is not given.
- **Filter out data** that is not desired in the destination system. Sub setting or removing observations based on some condition.
- **Check validity of data and fixing records**: ignore or report on rows that would cause an error, remove unwanted characters and duplicates.
- **Splitting and resampling**
- **Normalise/standardizing data** to remove variations that should be the same, e.g. replace upper case with lower case, replace "01" with "1".

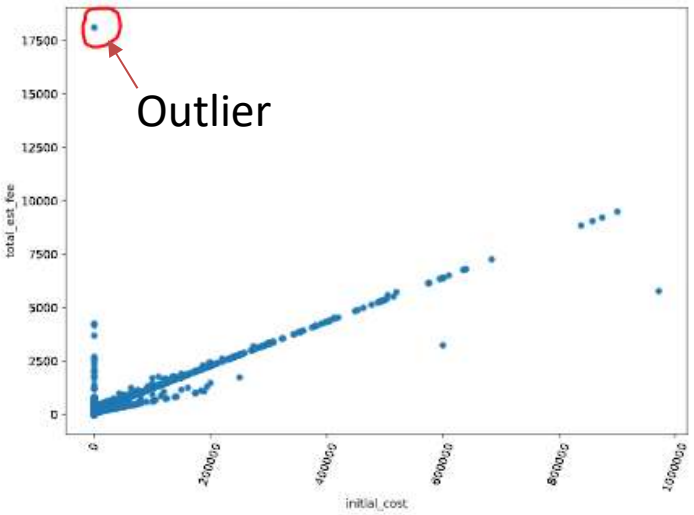
# Common tasks



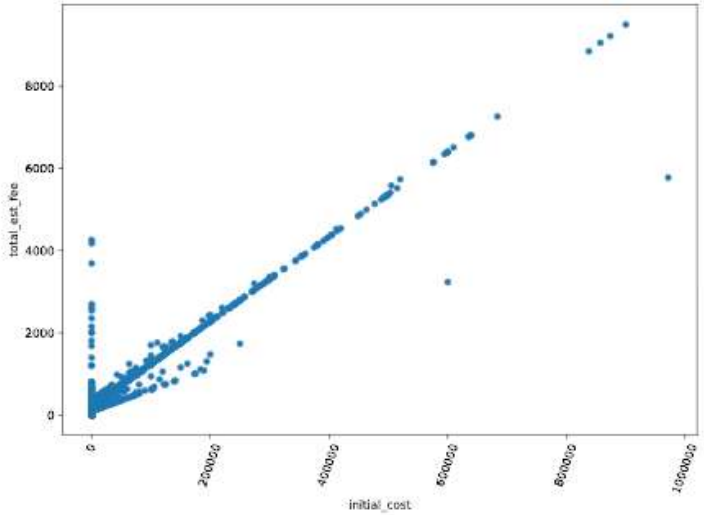

Design choices for reducing (or increasing) the amount of data items and attributes to show.

Tamara Munzner

# Common tasks



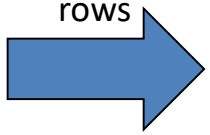

Removing the outlier



# Common tasks

| | Name | Height | Roll |
|---|---|---|---|
| 0 | A | 5.2 | 55 |
| 1 | A | 5.2 | 55 |
| 2 | C | 5.6 | 15 |
| 3 | D | 5.5 | 80 |
| 4 | E | 5.3 | 12 |
| 5 | E | 5.3 | 12 |
| 6 | G | 5.6 | 47 |
| 7 | H | 5.5 | 104 |

Removing duplicate rows



| | Name | Height | Roll |
|---|---|---|---|
| 0 | A | 5.2 | 55 |
| 2 | C | 5.6 | 15 |
| 3 | D | 5.5 | 80 |
| 4 | E | 5.3 | 12 |
| 6 | G | 5.6 | 47 |
| 7 | H | 5.5 | 104 |

# Potential activity transforming your data

**‘Before you can plot or graph anything, you have to find the data, understand it, evaluate it, clean it, and perhaps restructure it.’** (Marcia Gray, graphic designer)

**Three different types of potential activity involved in transforming your data:**

- **Cleaning:** resolve any data condition issues
- **Creating:** consider developing new calculations and value conversions
- **Consolidating:** think about introducing further data to expand or append to what you already have

# Potential activity transforming your data

- **Cleaning:** There is no single approach for how best to conduct data cleaning- Issues may be resolved through manual intervention, sorting, filtering, isolating, modifying any problem values/characters.

- **Creating:** Expand your data to form new calculations and derive new groupings or any other mathematical treatments. This may include:
  - Creating percentage calculations based on existing quantities.
  - Using ‘start date’ and ‘end date’ values to calculate the duration in days.
  - Using logic-based formulae to create new categorical values out of quantities
  - To derive reasonable categorical or quantitative values from the original form.

- **Consolidating:** you may seek to source and introduce additional data to **expand** (more variables) or **append** (more items) your data further in order to enhance its analytical potential

# Best practices

The questions you need to ask of your data are:

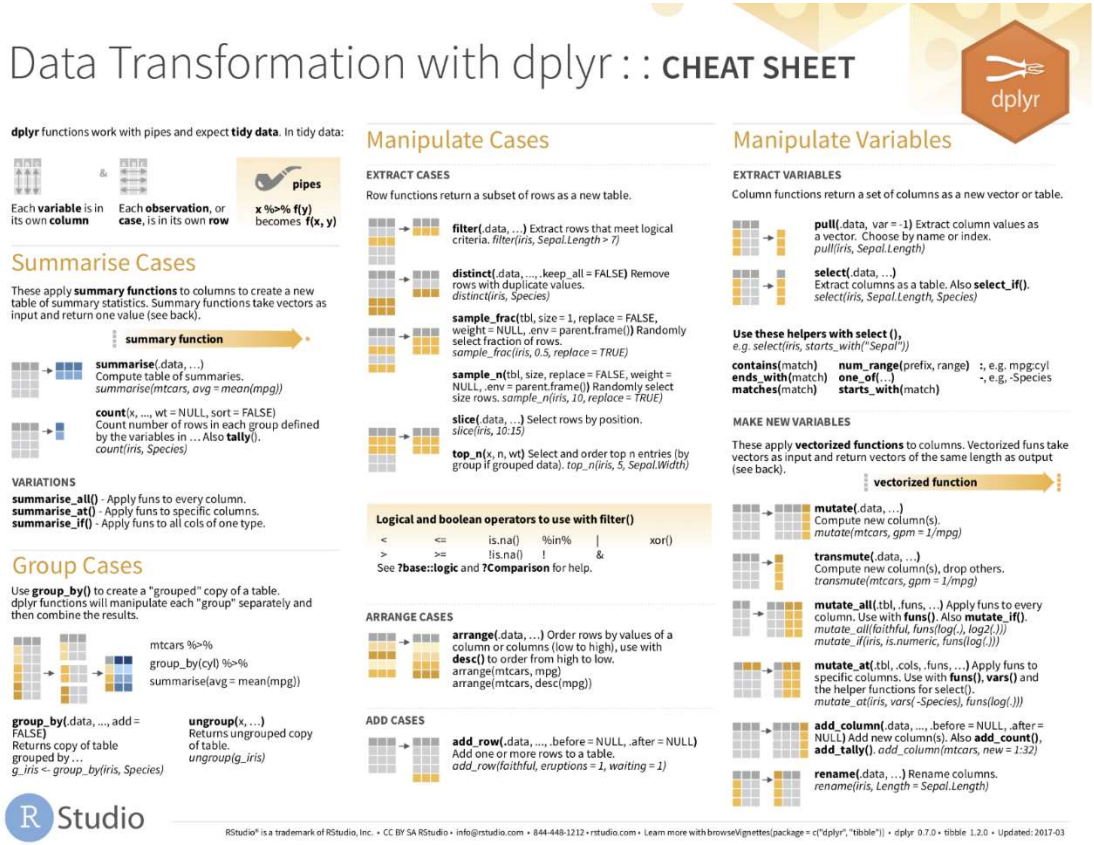
- Does it represent genuine observations about a given phenomenon or is it influenced by the limitations of a collection method?
- Does your data reflect the entirety of a particular phenomenon, a recognised sample, or maybe even an obstructed view caused by hidden limitations in the availability of data about that phenomenon?

**Once you complete your examination of your data you will have a good idea about what actions may be needed to transform your data.**

In accordance with the desire for trustworthy design, **any modifications or enhancements you apply to your data need to be noted and potentially explained to the people you show it**.

Kirk, A. (2016). *Data visualisation: A handbook for data driven design*. Sage.

# Data Transforming in R



Data Transformation with dplyr : : CHEAT SHEET

dplyr

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

Each variable is in its own column. Each observation, or case, is in its own row.

pipes: x %>% f(y) becomes f(x, y)

## Summarise Cases

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

summary function

summarise(.data, …) Compute table of summaries. *summarise(mtcars, avg = mean(mpg))*

count(x, ..., wt = NULL, sort = FALSE) Count number of rows in each group defined by the variables in … Also tally(). *count(iris, Species)*

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

## Group Cases

Use group_by() to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))

group_by(.data, ..., add = FALSE) Returns copy of table grouped by … *g_iris <- group_by(iris, Species)*

ungroup(x, …) Returns ungrouped copy of table. *ungroup(g_iris)*

## Manipulate Cases

EXTRACT CASES

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

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values. *distinct(iris, Species)*

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows. *sample_frac(iris, 0.5, replace = TRUE)*

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position. *slice(iris, 10:15)*

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

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

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

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with desc() to order from high to low. *arrange(mtcars, mpg)* *arrange(mtcars, desc(mpg))*

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table. *add_row(faithful, eruptions = 1, waiting = 1)*

## Manipulate Variables

EXTRACT VARIABLES

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

pull(.data, var = -1) Extract column values as a vector. Choose by name or index. *pull(iris, Sepal.Length)*

select(.data, …) Extract columns as a table. Also select_if(). *select(iris, Sepal.Length, Species)*

Use these helpers with select (), *e.g. select(iris, starts_with("Sepal"))*

contains(match) num_range(prefix, range) :, e.g. mpg:cyl
ends_with(match) one_of(…) -, e.g, -Species
matches(match) starts_with(match)

MAKE NEW VARIABLES

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

vectorized function

mutate(.data, …) Compute new column(s). *mutate(mtcars, gpm = 1/mpg)*

transmute(.data, …) Compute new column(s), drop others. *transmute(mtcars, gpm = 1/mpg)*

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with funs(). Also mutate_if(). *mutate_all(faithful, funs(log(.), log2(.)))* *mutate_if(iris, is.numeric, funs(log(.)))*

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with funs(), vars() and the helper functions for select(). *mutate_at(iris, vars(-Species), funs(log(.)))*

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also add_count(), add_tally(). *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns. *rename(iris, Length = Sepal.Length)*

RStudio



Look the data massage cheat sheats available

Barcelona Activa

30 ANYS

**barcelona.cat/barcelonactiva**