Data Handling: Import, Cleaning and Visualisation

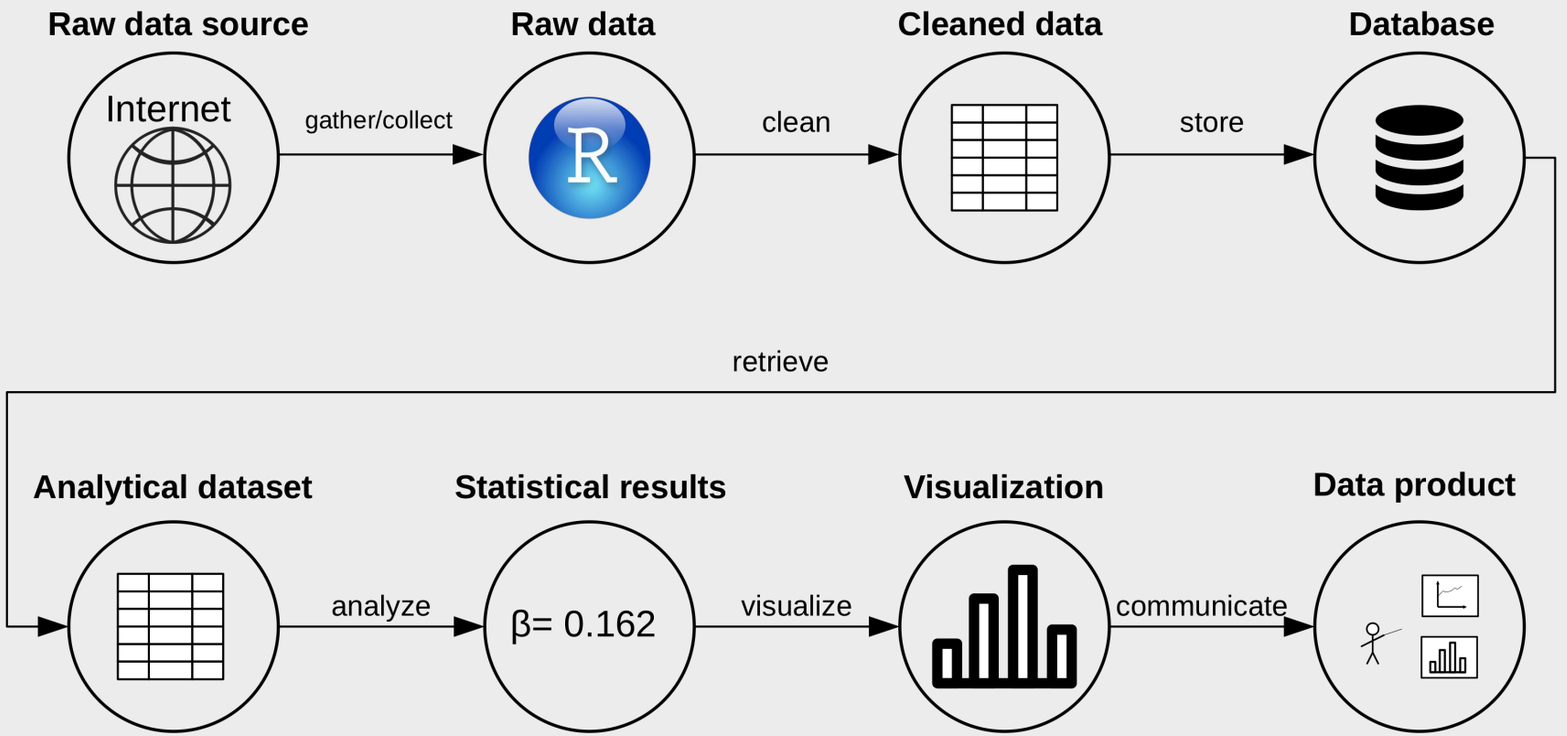
Lecture 8:
Basic Data Manipulation with R

Dr. Aurélien Sallin

2024-11-21

Recap: Data Preparation

Data (science) pipeline



Data preparation/data cleaning

Goal of data preparation: Dataset is ready for analysis.

Key conditions:

1. Data values are consistent/clean within each variable.
2. Variables are of proper data types.
3. Dataset is in ‘tidy’ (in long format)!

“Garbage in garbage out (GIGO)”

Data preparation consists of five main steps

- **Tidy** data.
- **Reshape** datasets from wide to long (and vice versa).
- **Bind** or stack rows in datasets.
- **Join** datasets.
- **Clean** data.

A tidy dataset is tidy, when ...

1. Each variable is a column; each column is a variable.
2. Each observation is a row; each row is an observation.
3. Each value is a cell; each cell is a single value.

Reshaping

country	year	metric
x	1960	10
x	1970	13
x	2010	15
y	1960	20
y	1970	23
y	2010	25
z	1960	30
z	1970	33
z	2010	35

```
pivot_wider(names_from = “year”,
            names_prefix = “yr”,
            values_from = “metric”)
```

country	yr1960	yr1970	yr2010
x	10	13	15
y	20	23	25
z	30	33	35

```
pivot_longer(cols = yr1960:yr2010,
             names_to = “year”,
             names_prefix = “yr”
             values_to = “metric”)
```

Long and wide data. Source: Hugo Tavares

Stack/row-bind

ID	X	Y
1	a	50
2	b	10

ID	Z
3	M
4	O

ID	X	Z
5	c	P

ID	X	Y	Z
1	a	50	NA
2	b	10	NA
3	NA	NA	M
4	NA	NA	O
5	c	NA	P

Warm up

Reshaping: multiple/one/none answers correct

Consider the following data frame `schwiizerChuchi`. This dataset records the popularity ratings (on a scale of 1 to 10) of various Swiss dishes in different regions of Switzerland:

```
schwiizerChuchi <- data.frame(
  Region = c("Zurich", "Geneva", "Lucerne"),
  Fondue = c(8, 9, 7),
  Raclette = c(7, 8, 10),
  Rosti = c(9, 6, 8),
  Olma = c(10, 7, 8)
)
```

```
schwiizerChuchiLong <- pivot_longer(schwiizerChuchi,
                                    cols = c(Fondue, Raclette, Rosti, Olma),
                                    values_to = "Popularity",
                                    names_to = "Dish")
```

Which of the following statements is true?

- `nrow(schwiizerChuchiLong) == 12` returns `TRUE`
- `dim(schwiizerChuchiLong)` returns `c(3, 12)`
- `dim(schwiizerChuchi)` returns `c(3, 12)`
- `mean(schwiizerChuchiLong$Raclette) == 8.333`

Tidy data: essay question

Why is this data frame not tidy, and what would you do to make it tidy? Write down your reasoning in numbered steps. You can write down some exact code, some higher-level code concepts, or in plain text.

```
temp_location_data <- data.frame(
  temperature_location = c("22C_London", "18C_Paris", "25C_Rome")
)
```

Tidy data: essay question

Why is this data frame not tidy, and what would you do to make it tidy? Write down your reasoning in numbered steps. You can write down some exact code, some higher-level code concepts, or in plain text.

```
grades_data <- data.frame(
  Student = c("Johannes", "Hannah", "Igor"),
  Econ = c(5, 5.25, 4),
  DataHandling = c(4, 4.5, 5),
  Management = c(5.5, 6, 6)
)
```

Today

Goals of today's lecture

1. Understand the concept of merging datasets.
2. Perform basic data manipulation in `dplyr`.
3. First steps in exploratory data analysis.

Data Preparation: merging

Merging (Joining) datasets

- Combine data of two datasets in one dataset.
- Needed: Unique identifiers for observations (‘keys’).

Merging (joining) datasets

x

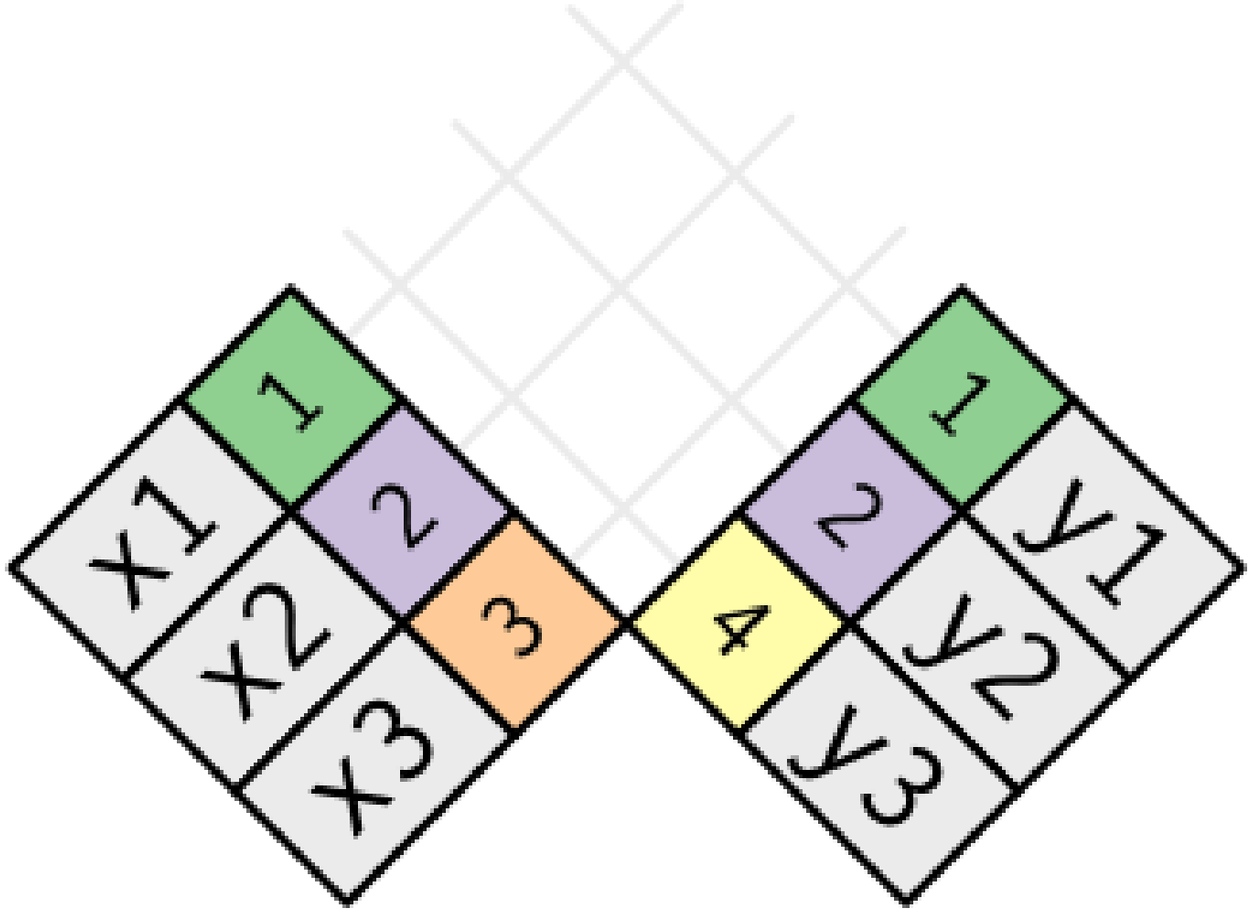
1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3

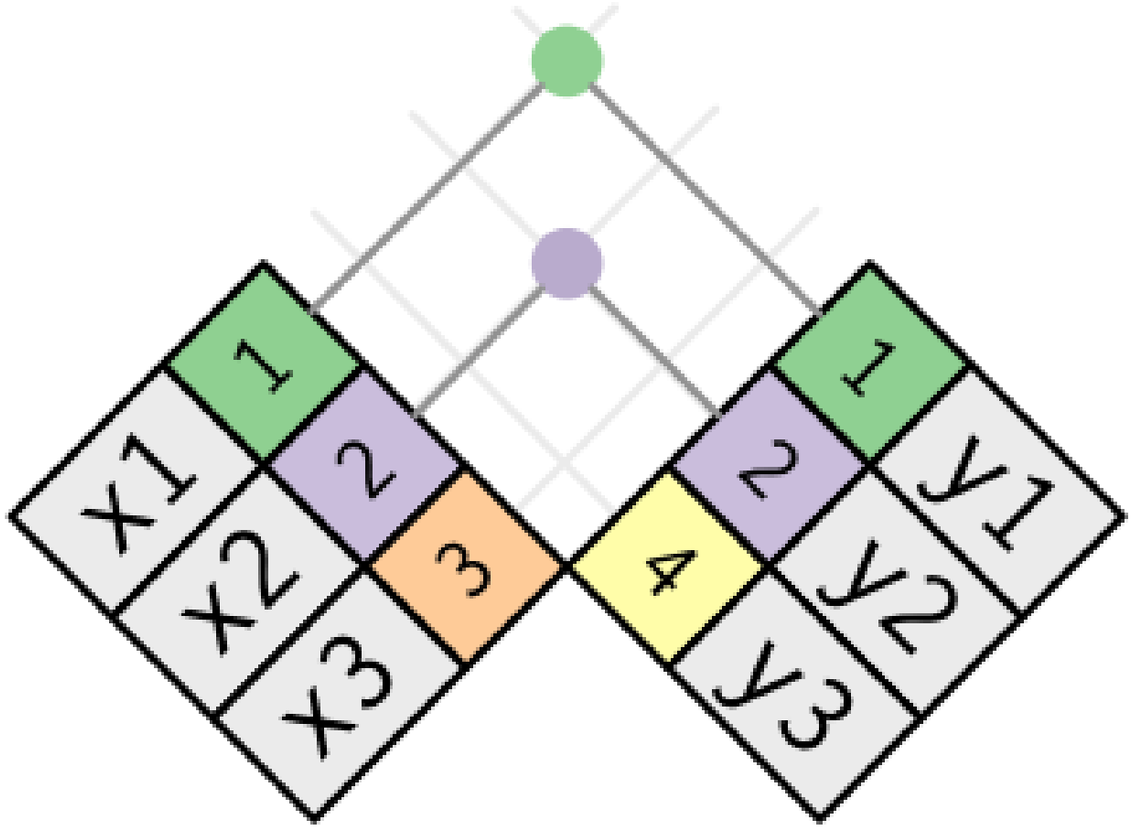
Join setup. Source: R4DS.

Merging (joining) datasets



Join setup. Source: R4DS.

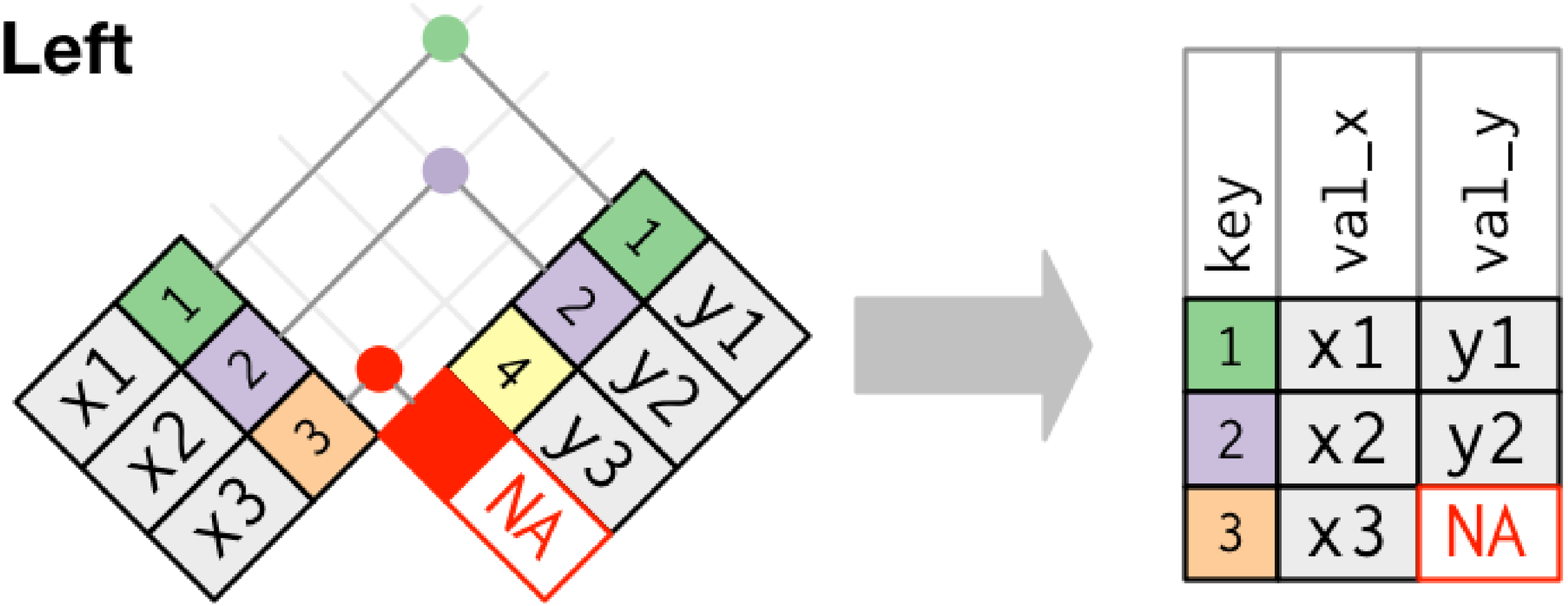
Merging (joining) datasets using an inner join



key	val_x	val_y
1	x1	y1
2	x2	y2

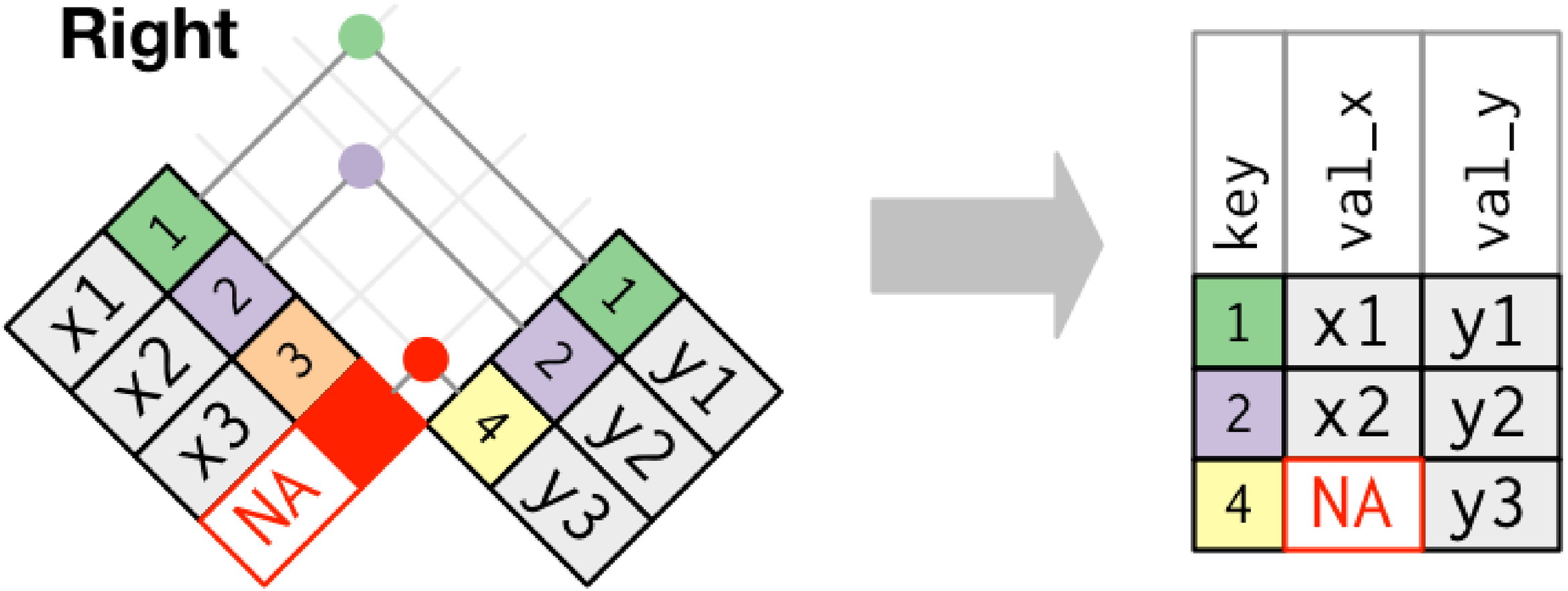
Inner join. Source: R4DS

Merging (joining) datasets using a left join



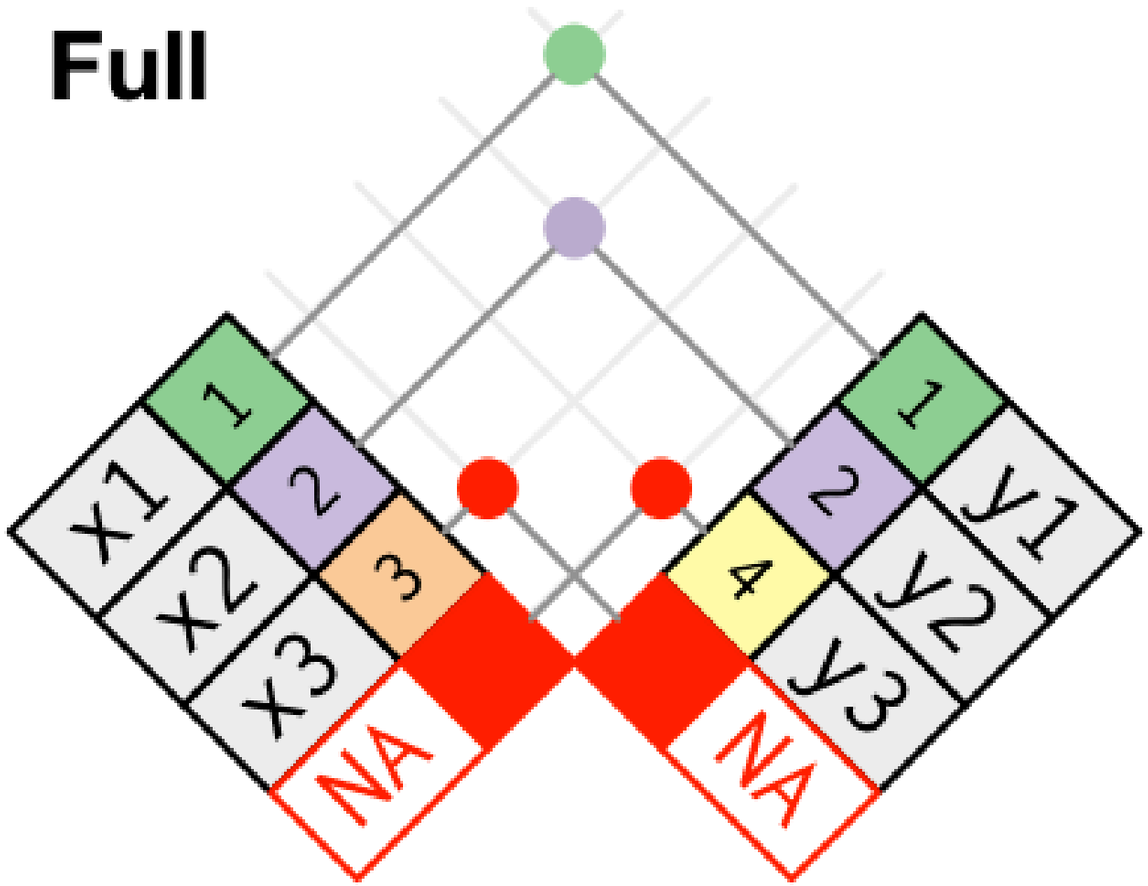
Left join. Source: R4DS.

Merging (joining) datasets using a right join



Right join. Source: R4DS.

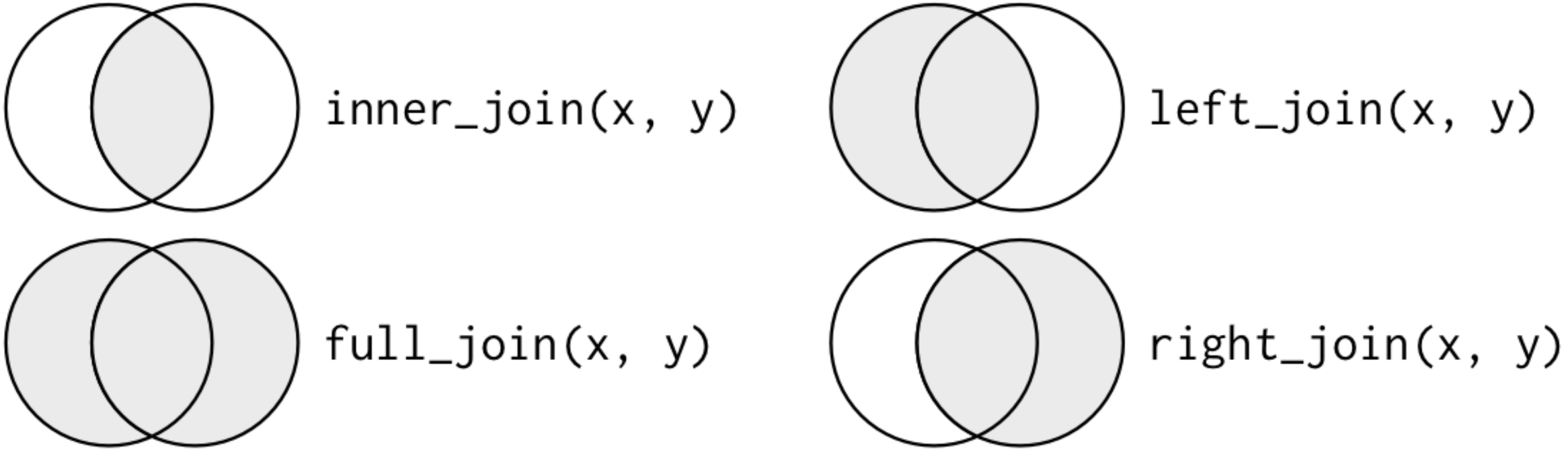
Merging all x and all y using a full join



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

Full join. Source: R4DS.

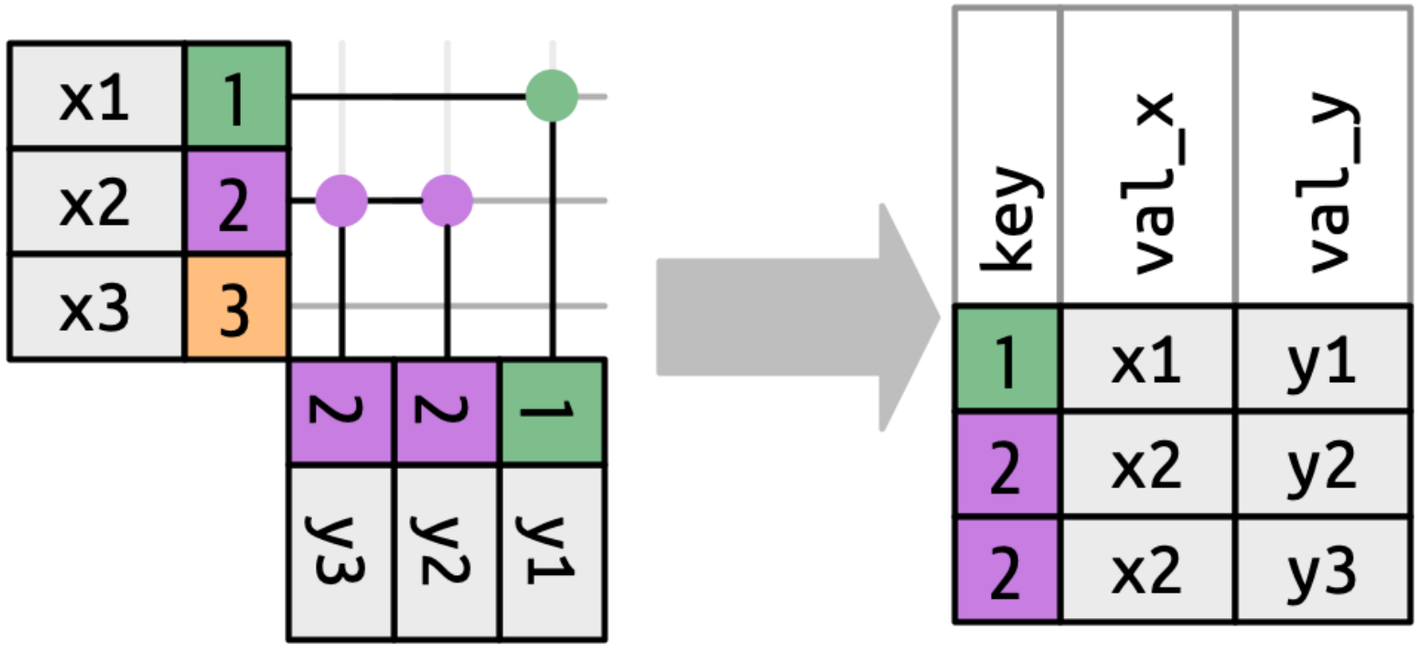
The four fundamental joins are :



Join Venn Diagramm. Source: R4DS.

Row-matching behaviors in joins

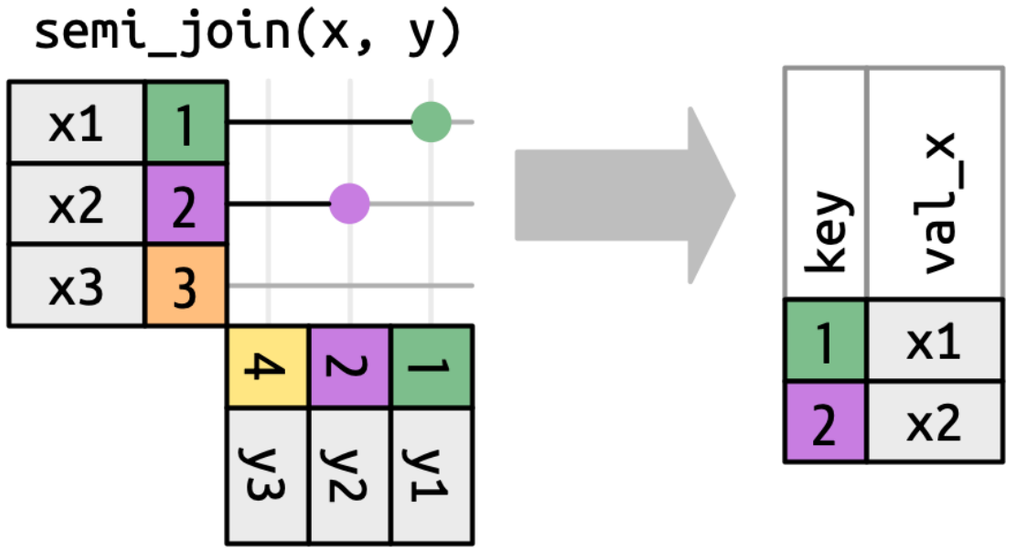
1. “One-to-one”
2. “Many-to-many”
3. “One-to-many”
4. “Many-to-one”



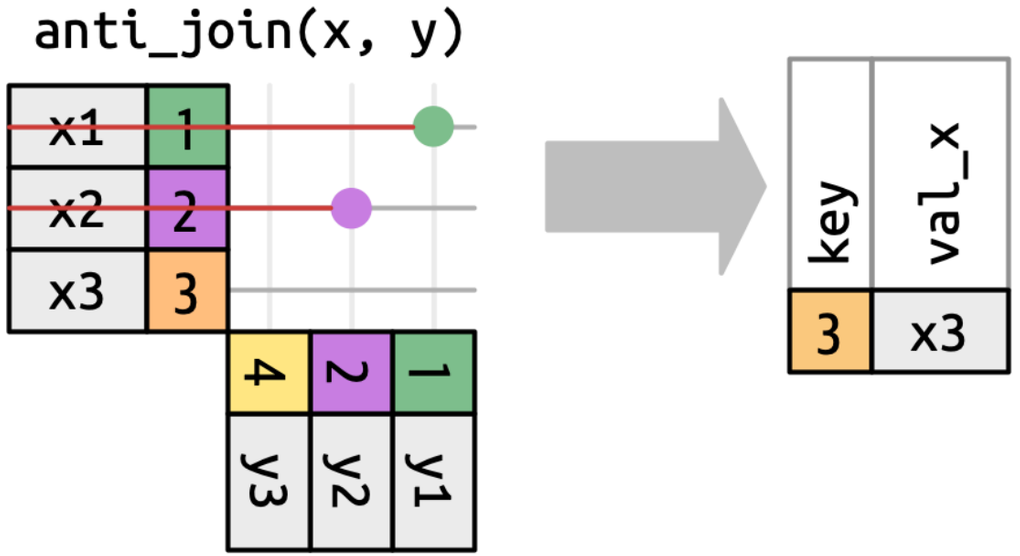
One-to-many joins. Source: R4DS

Always check how many rows are returned after your merge! In `tidyverse`, warnings appear in case of “many-to-many”. As of `dplyr 1.1.1`, no warning for one-to-many relationships.

Filtering joins with the semi join and the anti join



The semi-join keeps rows in x that have one or more matches in y. Source: R4DS.



The anti-join keeps rows in x that match zero rows in y. Source: R4DS.

Merging (joining) datasets: example

```
# load packages
library(tidyverse)

# initiate data frame on persons personal spending
df_c <- data.frame(id = c(1:3,1:3),
                   money_spent= c(1000, 2000, 6000, 1500, 3000, 5500),
                   currency = c("CHF", "CHF", "USD", "EUR", "CHF", "USD"),
                   year=c(2017,2017,2017,2018,2018,2018))
df_c
```

```
  id money_spent currency year
1  1        1000      CHF 2017
2  2        2000      CHF 2017
3  3        6000      USD 2017
4  1        1500      EUR 2018
5  2        3000      CHF 2018
6  3        5500      USD 2018
```

Merging (joining) datasets: example

```
# initiate data frame on persons' characteristics
df_p <- data.frame(id = 1:4,
                   first_name = c("Anna", "Betty", "Claire", "Diane"),
                   profession = c("Economist", "Data Scientist",
                                  "Data Scientist", "Economist"))
df_p
```

```
  id first_name     profession
1  1       Anna      Economist
2  2      Betty Data Scientist
3  3     Claire Data Scientist
4  4      Diane      Economist
```

Merging (joining) datasets: example

```
df_merged <- left_join(df_p, df_c, by="id")
df_merged
```

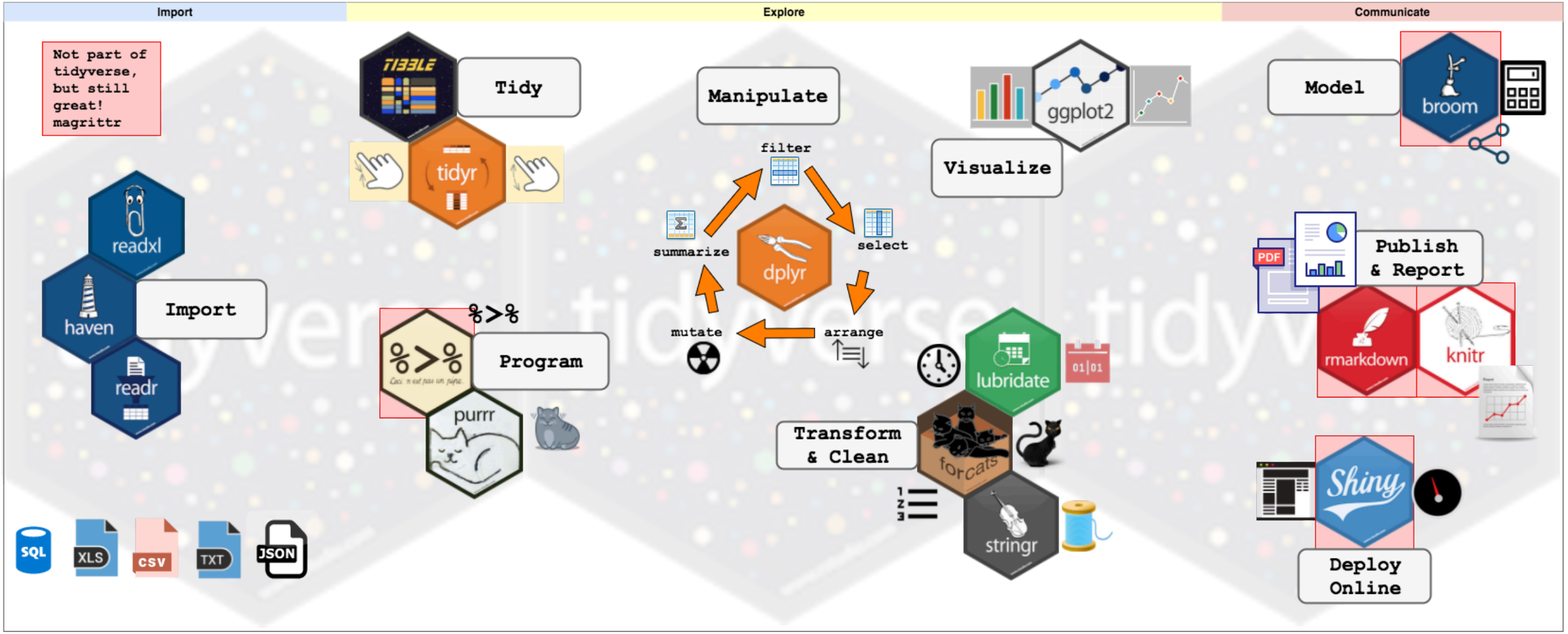
```
  id first_name     profession money_spent currency year
1  1       Anna      Economist        1000      CHF 2017
2  1       Anna      Economist        1500      EUR 2018
3  2      Betty Data Scientist        2000      CHF 2017
4  2      Betty Data Scientist        3000      CHF 2018
5  3     Claire Data Scientist        6000      USD 2017
6  3     Claire Data Scientist        5500      USD 2018
7  4      Diane      Economist          NA     <NA>   NA
```

Merging (joining) datasets: R

Overview by R4DS:

dplyr (tidyverse)	base::merge
`inner_join(x, y)`	`merge(x, y)`
`left_join(x, y)`	`merge(x, y, all.x = TRUE)`
`right_join(x, y)`	`merge(x, y, all.y = TRUE),`
`full_join(x, y)`	`merge(x, y, all = TRUE)`

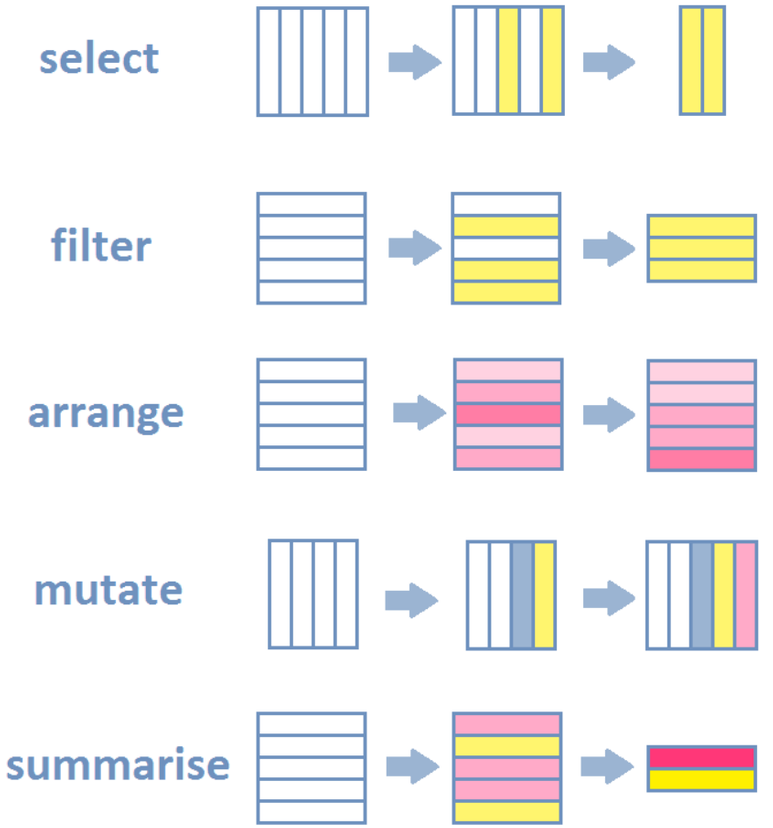
Transforming and cleaning data



Source: https://www.storybench.org/wp-content/uploads/2017/05/tidyverse.png

`select`, `filter`, `arrange`, `mutate` are the building blocks of `dplyr`

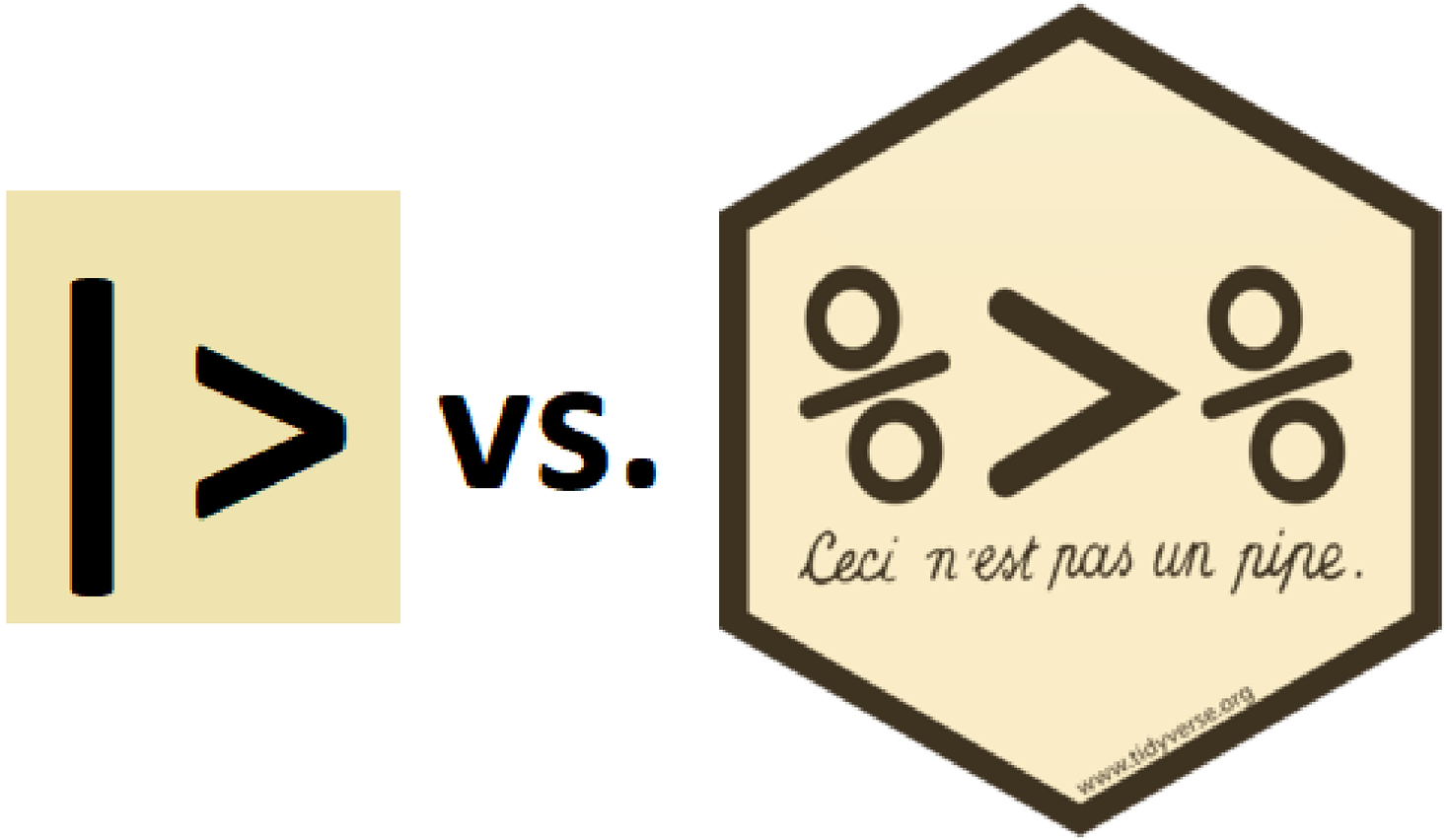
- **Select** the subset of variables you need (e.g., for comparisons).
- **Filter** the dataset by restricting your dataset to observations needed in *this* analysis.
- **Arrange** the dataset by reordering the rows.
- **Mutate** the dataset by adding the values you need for your analysis.
- **Group** and **summarize** the dataset by a variable to apply functions to groups of observations.



Source: Intro to R for Social Scientists

Prepare your data in a `pipe`line

- The operator has been now replaced with `|>`.



Prepare your data in a pipeline with dplyr

```r
# Traditional way
mydf <- data(swiss)
mydf <- arrange(mydf, -Catholic)
mydf <- filter(mydf, Education > 8 & Catholic > 90)
mydf <- mutate(mydf, Country = "Switzerland")
mydf <- select(mydf, Examination)

# The pipe way
mydf <- data(swiss)  |>
  arrange(-Catholic) |>
  filter(Education > 8 & Catholic > 90) |>
  mutate(Country = "Switzerland") |>
  select(Examination)
```

```r
# Base-R equivalent
mydf <- data(swiss)
mydf <- mydf[order(-mydf$Catholic), ]
mydf <- mydf[mydf$Education > 8 & mydf$Catholic > 90, ]
mydf$Country <- "Switzerland"
mydf <- mydf["Examination"]
```

Further tools for data transformation and cleaning in `dplyr`

- `forecats` to deal with factors;
- `lubridate` to deal with dates;
- `stringr` to deal with strings and regular expressions.