Relational data

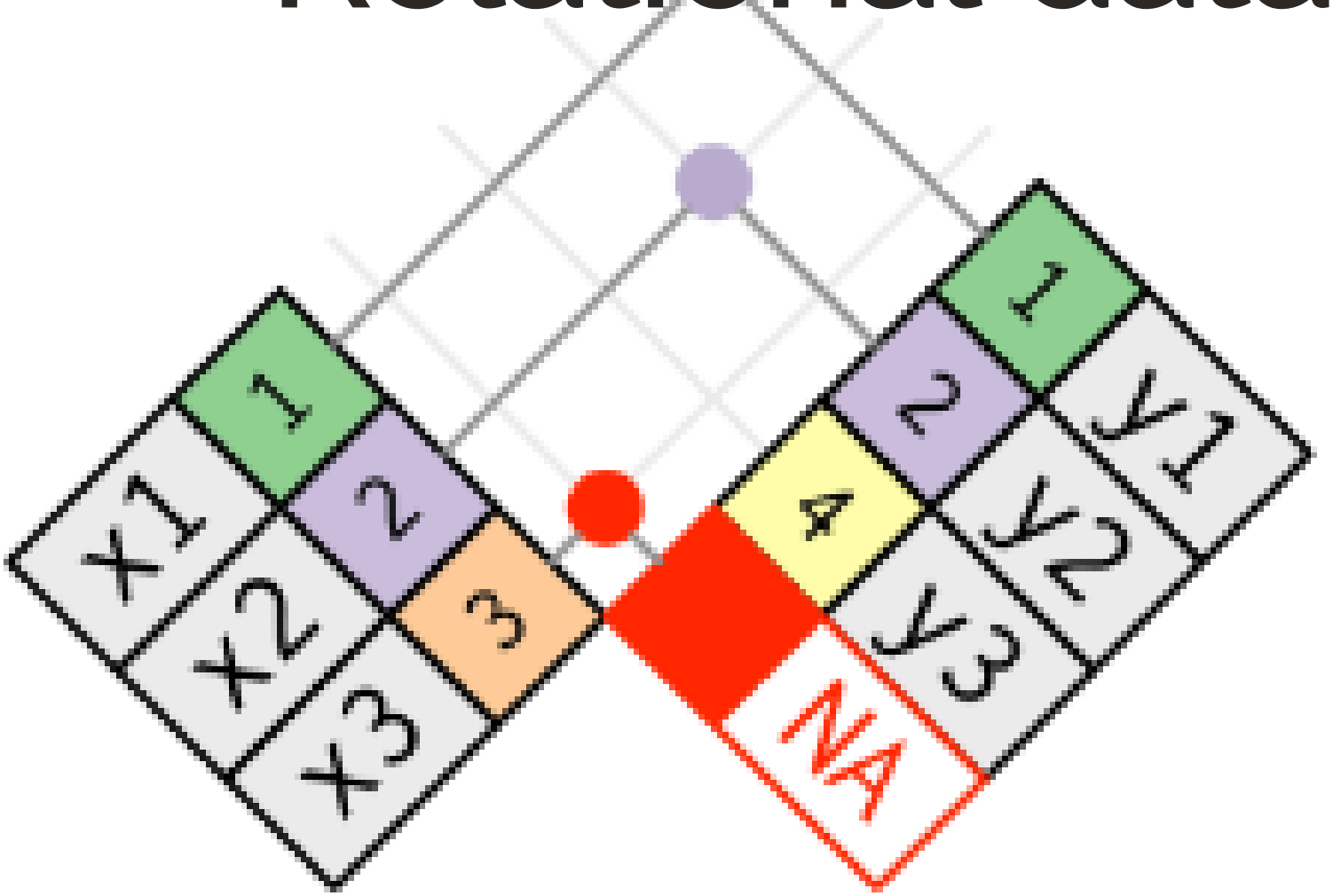


Image: R for Data Science

Relational data

Rare for all data to be found in one table.

A "key" variable is found in both tables, but val_x and val_y are separate.

table_1

key	val_x
1	x1
2	x2
3	x3

table_2

key	val_y
1	y1
2	y2
4	NA

Images: R for Data Science

Relational data

Here, we'll focus on left (outer) joins. The syntax is similar for other types of join.

table_1

key	val_x
1	x1
2	x2
3	x3

table_2

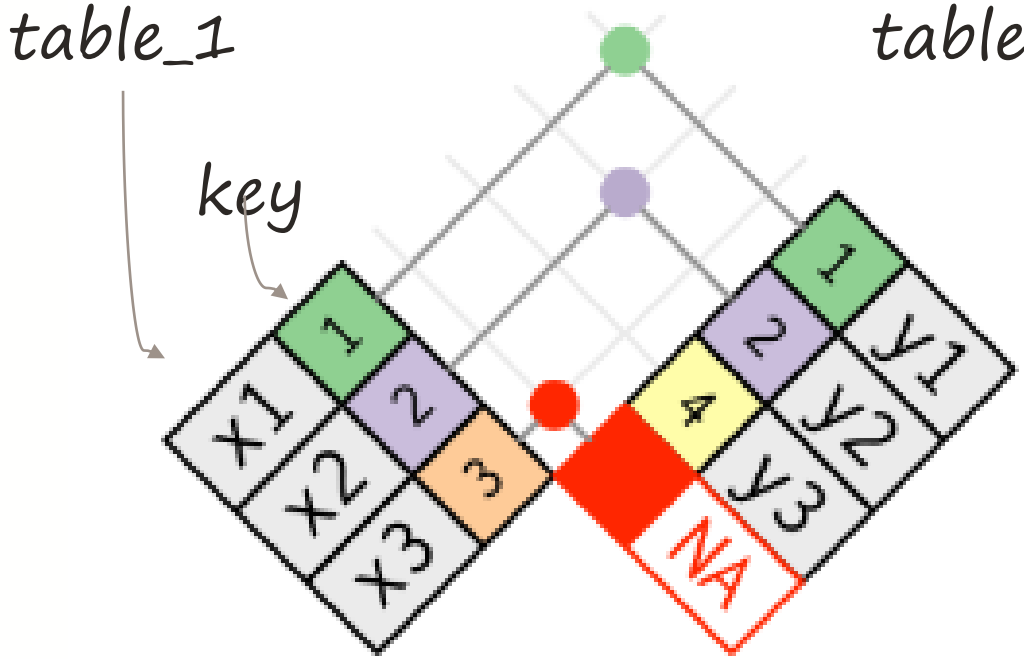
key	val_y
1	y1
2	y2
4	NA

Images: R for Data Science

left_join (left outer join)

table_1

table_2



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

Keep structure of table_1 and match to rows in table_2 based on "key" value

Images: R for Data Science

left_join

Keep structure of table_1

```
table_1 %>%
  left_join(table_2, by = "key")
```

...and match to rows in table_2

based on "key" value

Relational Data

We're going to join two tables - one with cases of tuberculosis by country, one with population by country. From this new table we can derive a rate.

cases

key	val_x
1	x1
2	x2
3	x3

pop

key	val_y
1	y1
2	y2
4	NA

W.H.O. data

Keep structure of cases

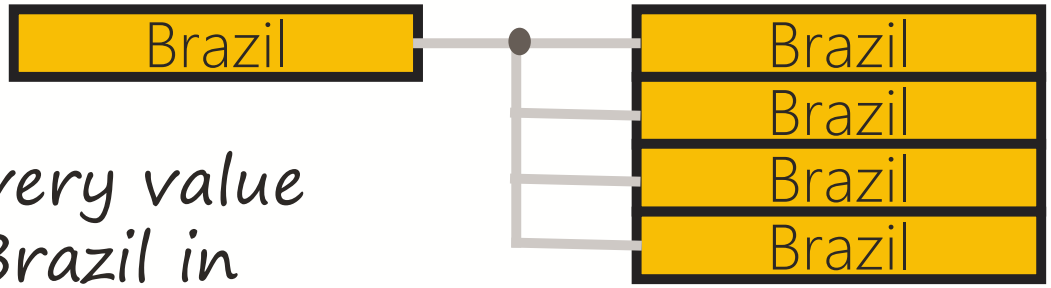
```
cases %>%
  left_join(pop, by = "country")
```

...then match to rows in pop

based on "country" value

Duplicates

```
cases %>%
  left_join(pop, by = “country”)
```



For every value of Brazil in tb_cases, there are 4 in pop

Joining on multiple rows

match on two variables

```
cases %>%
    left_join(pop, by = c("country" , "year"))
```

Joining with different names

Two tables have different name for same variable:

```
tb_cases %>%
    left_join(bad_names,
    by = c("country" = "Place" , "year" = "Yr"))
```

name in cases

name in bad_names

Some other dplyr joins

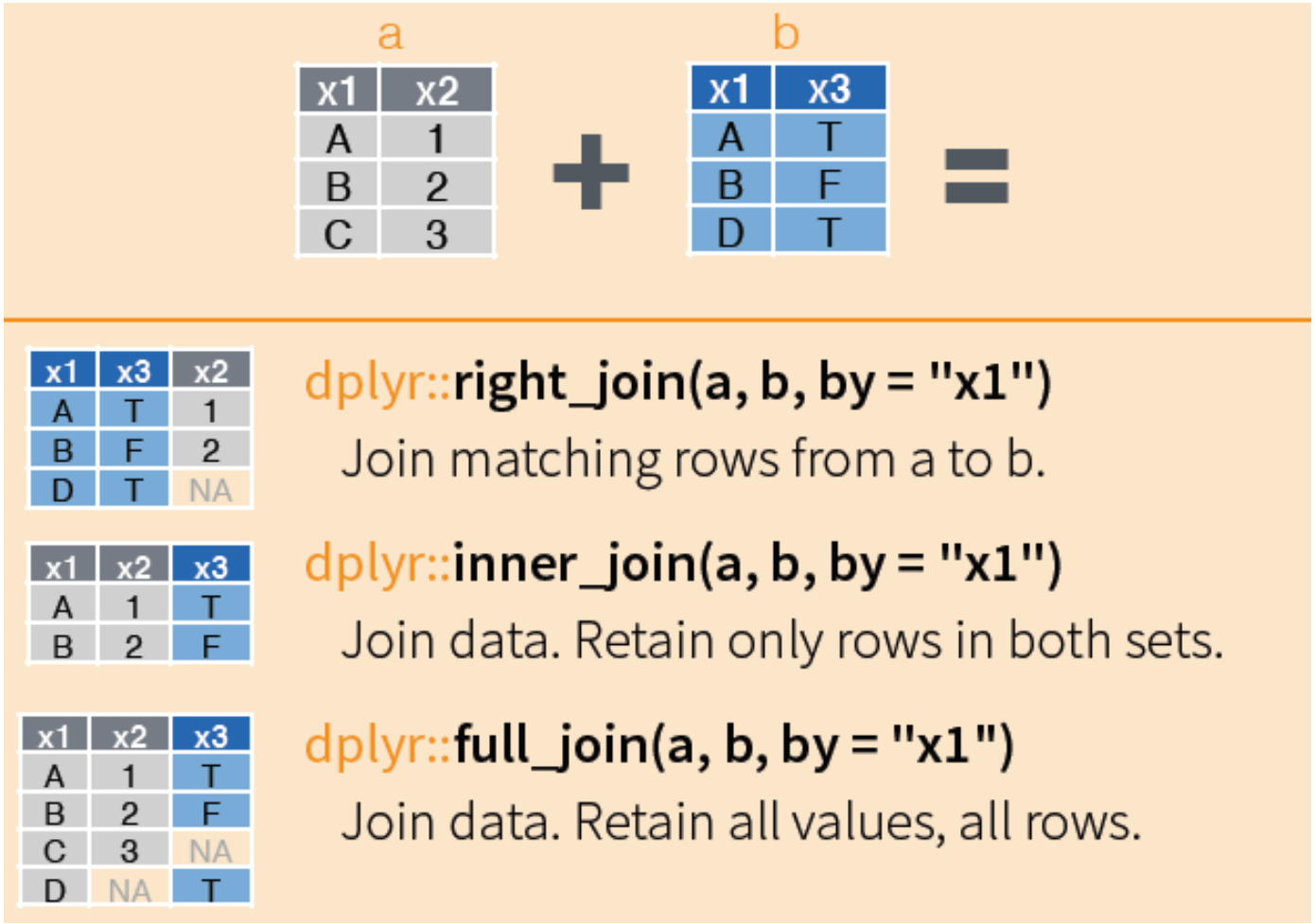


Image taken from: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

This work is licensed as

End