Tidy Data and Data Joins

Week 7 (11/12/25)

Artwork by Allison Horst

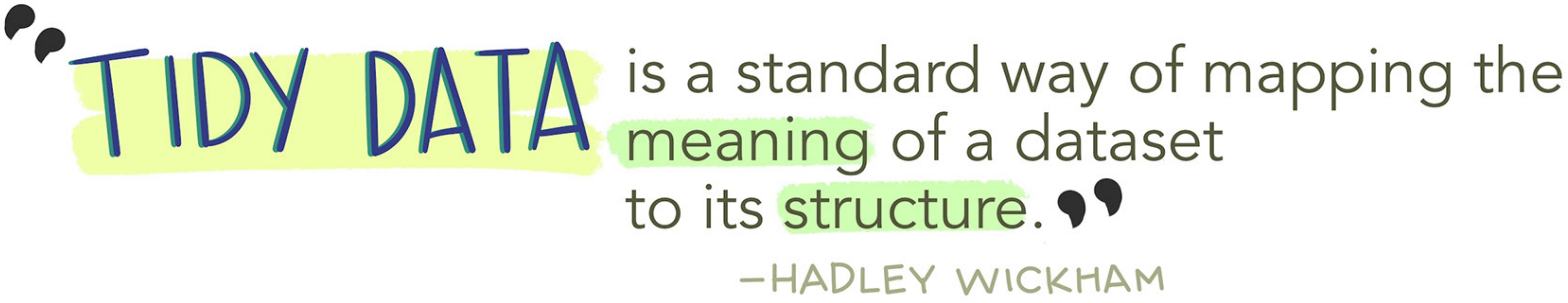
Stepfanie M. Aguillon

Outline of today's class

- What is tidy data?
- Introduction to tidying data with `tidyr`
- Introduction to joining datasets with `dplyr`

What is tidy data?

What is tidy data?



variables

observations

values

R for Data Science

What is tidy data?

"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

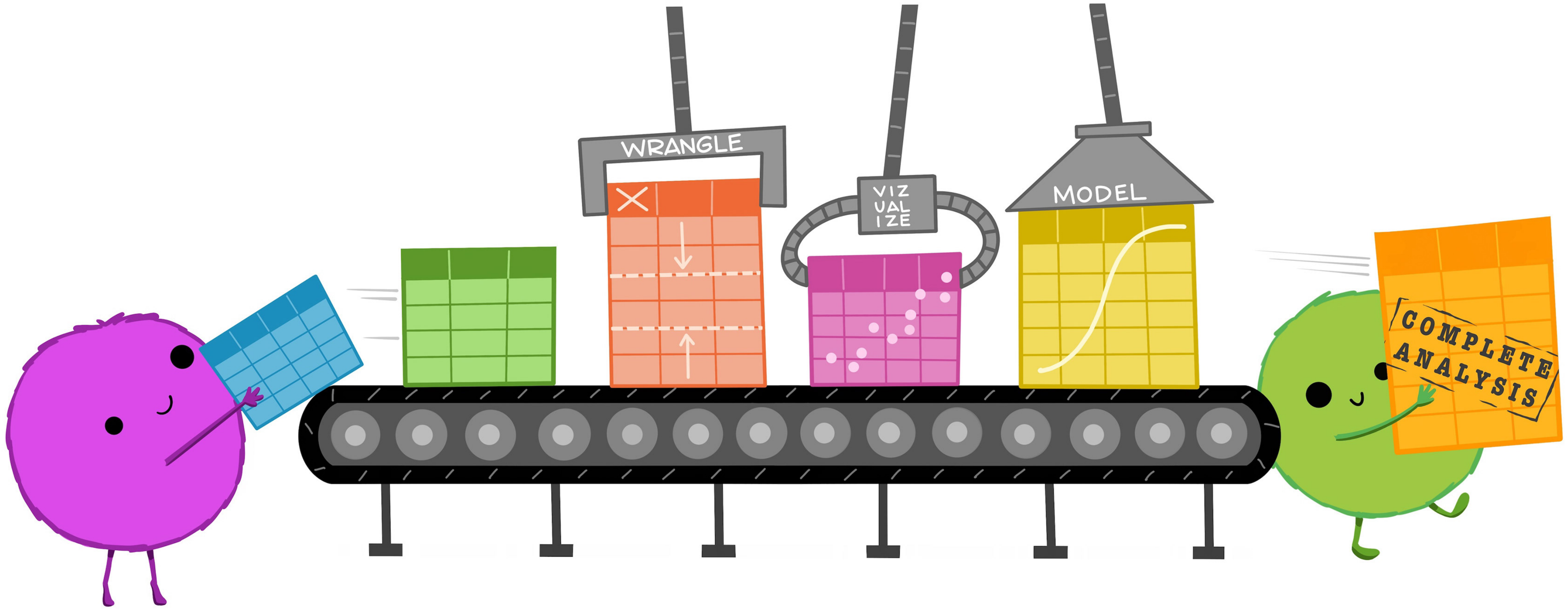
each row an observation

Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

Artwork by Allison Horst

So... why use tidy data?

"Tidy datasets are all alike, but every messy dataset is messy in its own way." - Hadley Wickham



Artwork by Allison Horst

R for Data Science

Advantages to tidy data

1. With a consistent tidy data structure, tools you use or develop with one dataset will work with others.

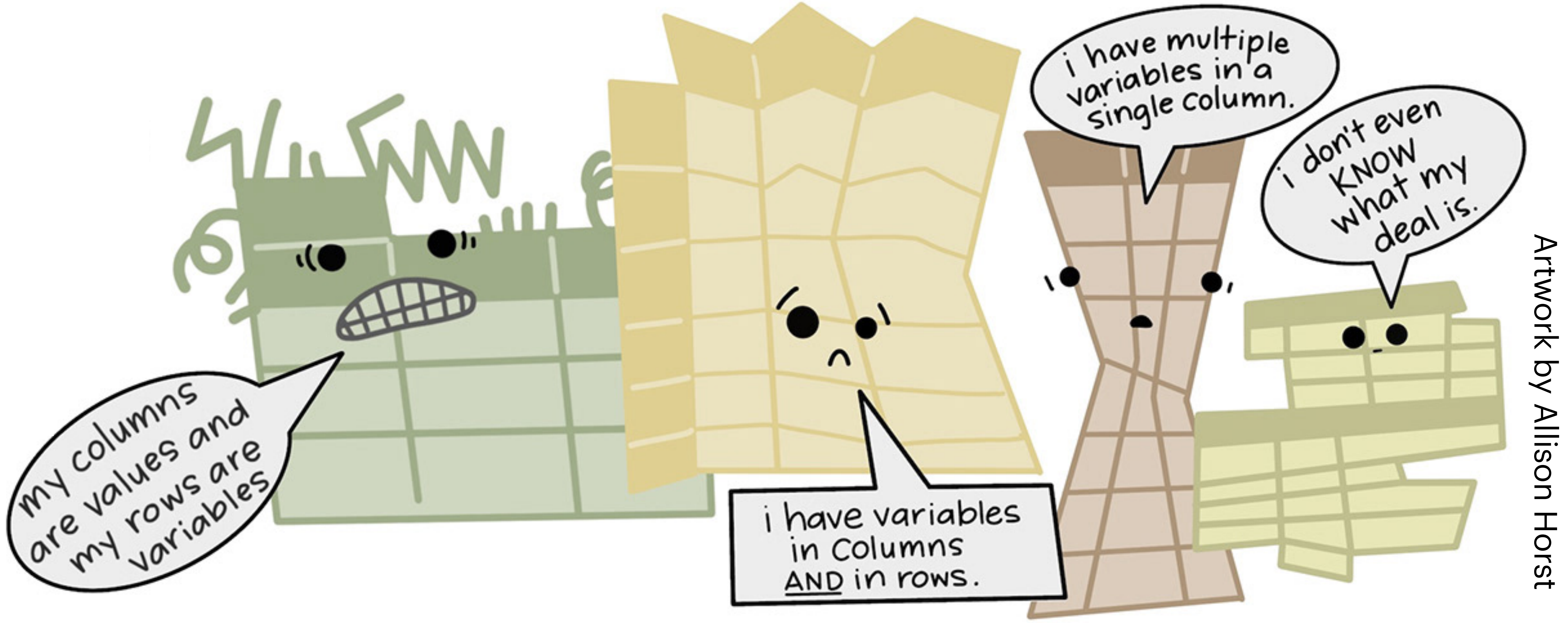
2. R really likes vectors! Storing your variables as columns allows R to work with them as vectors more easily.

3. The `tidyverse` is designed to work with tidy data.

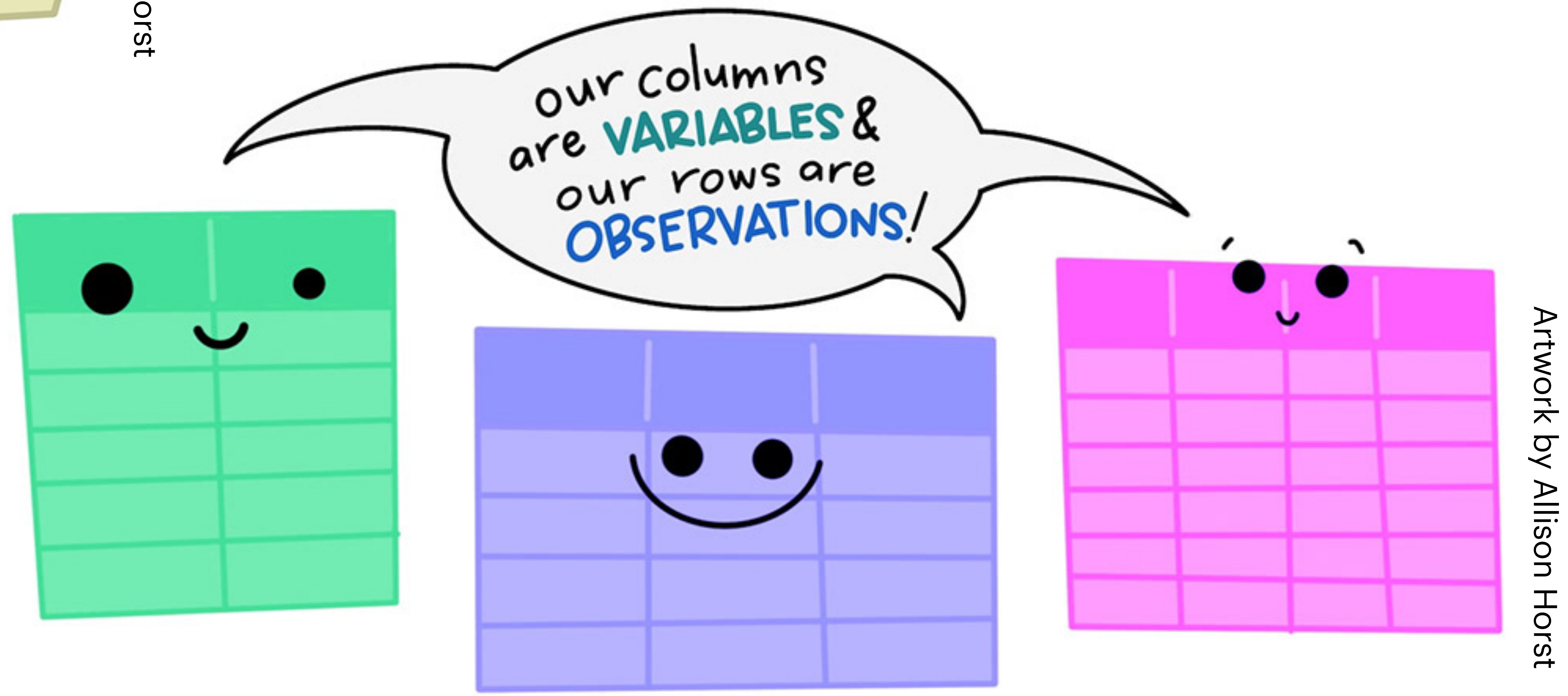
Human readable data ≠ Tidy data

film_number	Dwarf	Elf	Hobbit	Man	Orc	Wizard
I	431	2200	3658	1995	27	2881
II	521	844	2463	3990	230	1273
III	313	693	2675	2727	466	1807

Human readable data ➔ Tidy data

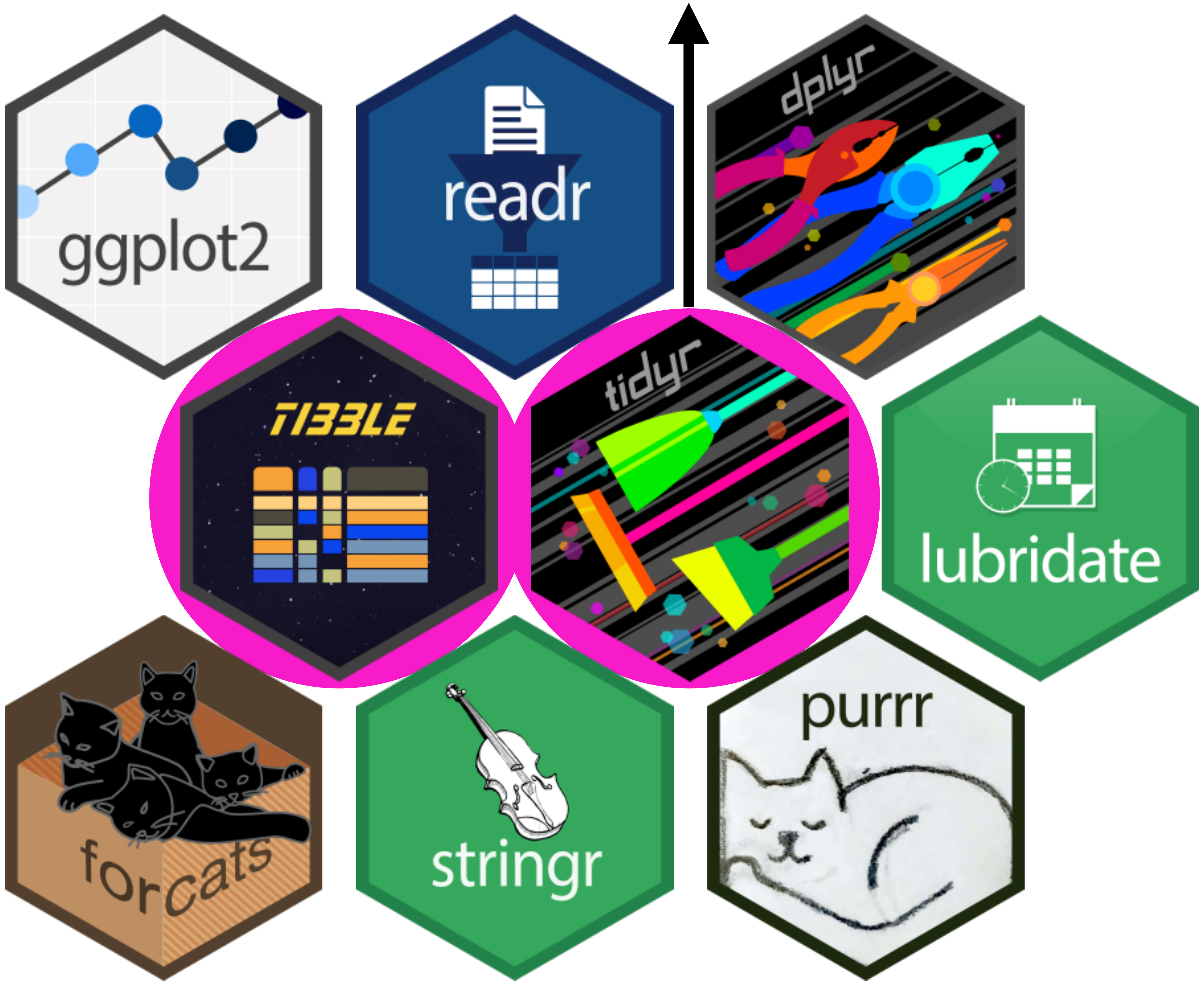


use the `tidyverse`



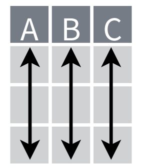
Tidying data with `tidyr`

`tidyverse` package
for tidying data



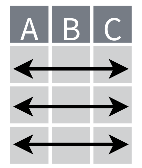
Data tidying with tidyr : : CHEATSHEET

Tidy data is a way to organize tabular data in a consistent data structure across packages.
A table is tidy if:

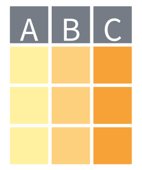


Each **variable** is in its own **column**

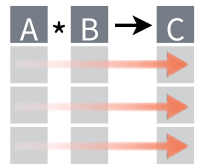
&



Each **observation**, or **case**, is in its own row



Access **variables** as **vectors**



Preserve **cases** in vectorized operations

Tibbles

AN ENHANCED DATA FRAME

Tibbles are a table format provided by the **tibble** package. They inherit the data frame class, but have improved behaviors:

- **Subset** a new tibble with], a vector with [[and $.
- **No partial matching** when subsetting columns.
- **Display** concise views of the data on one screen.

options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)** Control default display settings.

View() or **glimpse()** View the entire data set.

CONSTRUCT A TIBBLE

tibble(…**)** Construct by columns.
tibble(x = 1:3, y = c("a", "b", "c"))

tribble(…**)** Construct by rows.
tribble(~x, ~y,
1, "a",
2, "b",
3, "c")

Both make this tibble

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

as_tibble(x, …**)** Convert a data frame to a tibble.

enframe(x, name = "name", value = "value"**)** Convert a named vector to a tibble. Also **deframe()**.

is_tibble(x**)** Test whether x is a tibble.

Reshape Data

- Pivot data to reorganize values into a new layout.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

pivot_longer(table4a, cols = 2:3, names_to ="year", values_to = "cases")

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

pivot_wider(data, names_from = "name", values_from = "value"**)**

The inverse of pivot_longer(). "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

pivot_wider(table2, names_from = type, values_from = count)

Split Cells

- Use these functions to split or combine cells into individual, isolated values.

table5

country	century	year
A	19	99
A	20	00
B	19	99
B	20	00

→

country	year
A	1999
A	2000
B	1999
B	2000

unite(data, col, …, sep = "_", remove = TRUE, na.rm = FALSE**)** Collapse cells across several columns into a single column.

unite(table5, century, year, col = "year", sep = "")

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174

separate_wider_delim(data, cols, delim, ..., names = NULL, names_sep = NULL, names_repair = "check unique", too_few, too_many, cols_remove = TRUE**)** Separate each cell in a column into several columns. Also **separate_wider_regex()** and **separate_wider_position()**.

separate(table3, rate, sep = "/", into = c("cases", "pop"))

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M

separate_longer_delim(data, cols, delim, .., width, keep_eampty**)** Separate each cell in a column into several rows.

separate_longer_delim(table3, rate, sep = "/")

Expand Tables

Create new combinations of variables or identify implicit missing values (combinations of variables not present in the data).

x

x1	x2	x3
A	1	3
B	1	4
B	2	3

→

x1	x2
A	1
A	2
B	1
B	2

expand(data, …**)** Create a new tibble with all possible combinations of the values of the variables listed in … Drop other variables.
expand(mtcars, cyl, gear, carb)

x

x1	x2	x3
A	1	3
B	1	4
B	2	3

→

x1	x2	x3
A	1	3
A	2	NA
B	1	4
B	2	3

complete(data, …, fill = list()**)** Add missing possible combinations of values of variables listed in … Fill remaining variables with NA.
complete(mtcars, cyl, gear, carb)

Handle Missing Values

Drop or replace explicit missing values (NA).

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(data, …**)** Drop rows containing NA's in … columns.
drop_na(x, x2)

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(data, …, .direction = "down"**)** Fill in NA's in … columns using the next or previous value.
fill(x, x2)

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(data, replace**)** Specify a value to replace NA in selected columns.
replace_na(x, list(x2 = 2))

CC BY SA Posit Software, PBC • info@posit.co • posit.co • Learn more at **tidyr.tidyverse.org** • HTML cheatsheets at **pos.it/cheatsheets** • tidyr 1.3.1 • tibble 3.3.0 • Updated: 2025–08

tidyr Cheatsheet

Types of datasets

WIDE

Wide format

ID	X	Y	Z
1	a	b	c
2	d	e	f

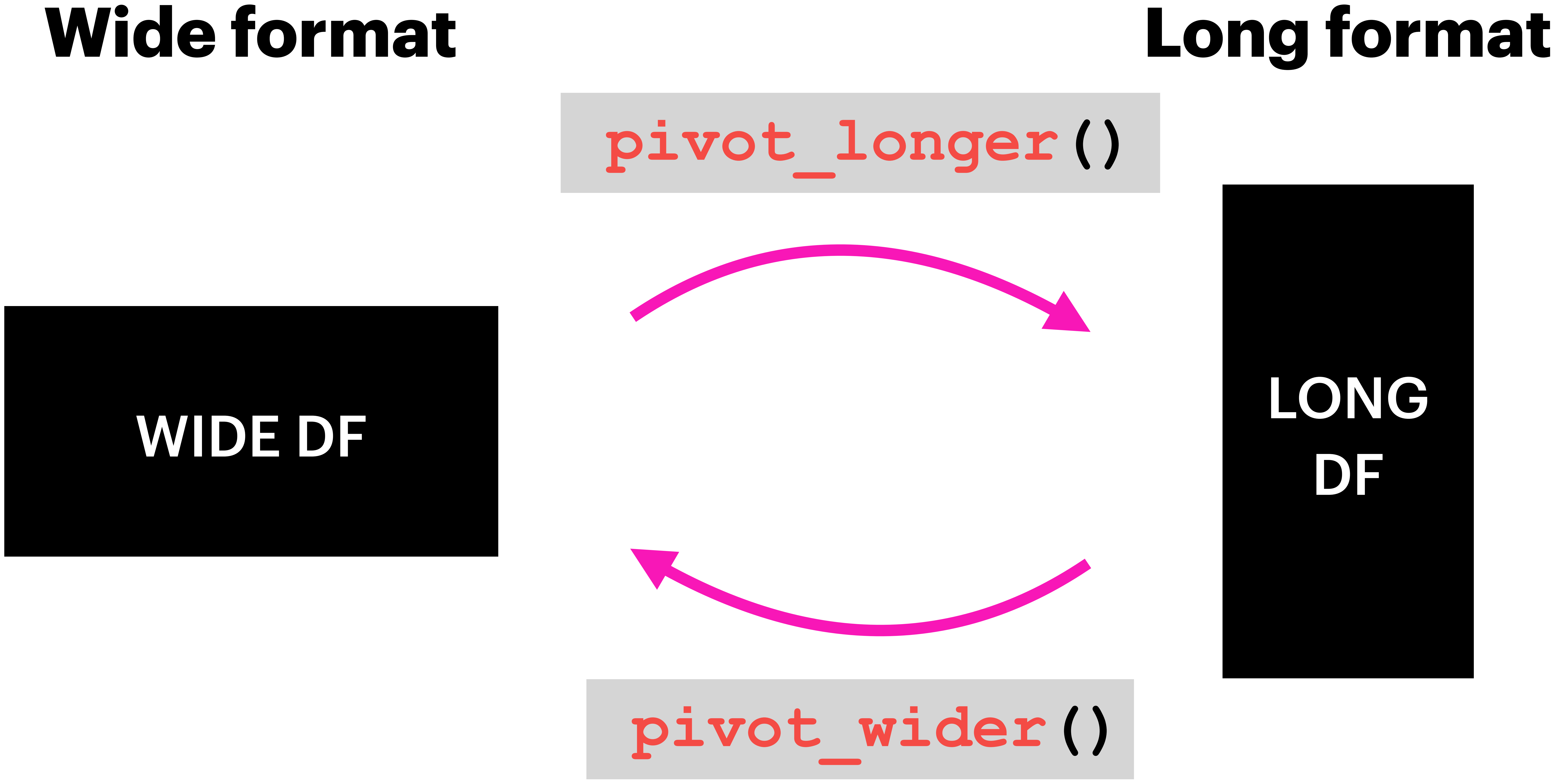
LONG

Long format

TIDY!

ID	variable	value
1	X	a
1	Y	b
1	Z	c
2	X	d
2	Y	e
2	Z	f

tidyr lets you move between formats



What are the names? What are the values?

```
pivot_longer(df, cols, names_to = “”, values_to = “”)
```

```
pivot_wider(df, names_from = , values_from = )
```

What are the names? What are the values?

```
pivot_longer(df, cols, names_to = “”, values_to = “”)
```

cols

names_to

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

values_to

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

What are the names? What are the values?

```
pivot_wider(df, names_from = , values_from =  )
```

names_from values_from

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M

What are the names? What are the values?

```
pivot_longer(df, cols, names_to = “”, values_to = “”)
```

`names_to`/`values_to` are the names we want for our new columns

```
pivot_wider(df, names_from = , values_from = )
```

`names_from`/`values_from` are the columns we’re getting info from

Practice, practice, practice...

Let's turn this into a tidy dataset!

`lotr_untidy_dataset.csv`

film_ number	Dwarf	Elf	Hobbit	Man	Orc	Wizard
I	431	2200	3658	1995	27	2881
II	521	844	2463	3990	230	1273
III	313	693	2675	2727	466	1807

Let's turn this into a tidy dataset!

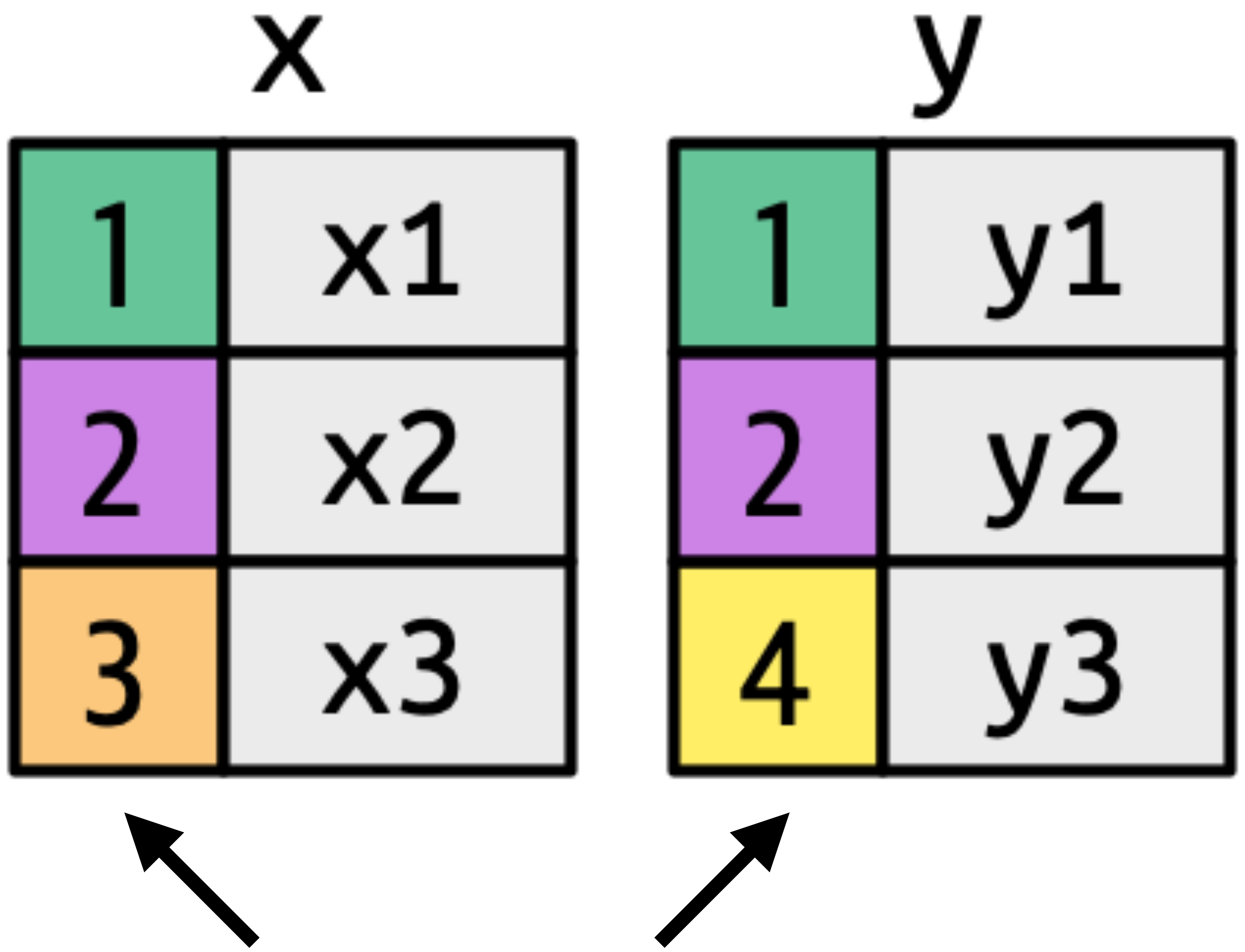
```
lotr_wide <- read_csv("lotr_untidy_dataset.csv")

lotr_long <- lotr_wide %>%
  pivot_longer(2:7, names_to = "race", values_to
  = "words_spoken")
```

Let's turn this into a tidy dataset!

film_number	race	words_spoken
I	Dwarf	431
I	Elf	2200
I	Hobbit	3658
I	Man	1995
I	Orc	27
I	Wizard	2881

Joining datasets with `dplyr`

But what if you have *multiple* datasets?



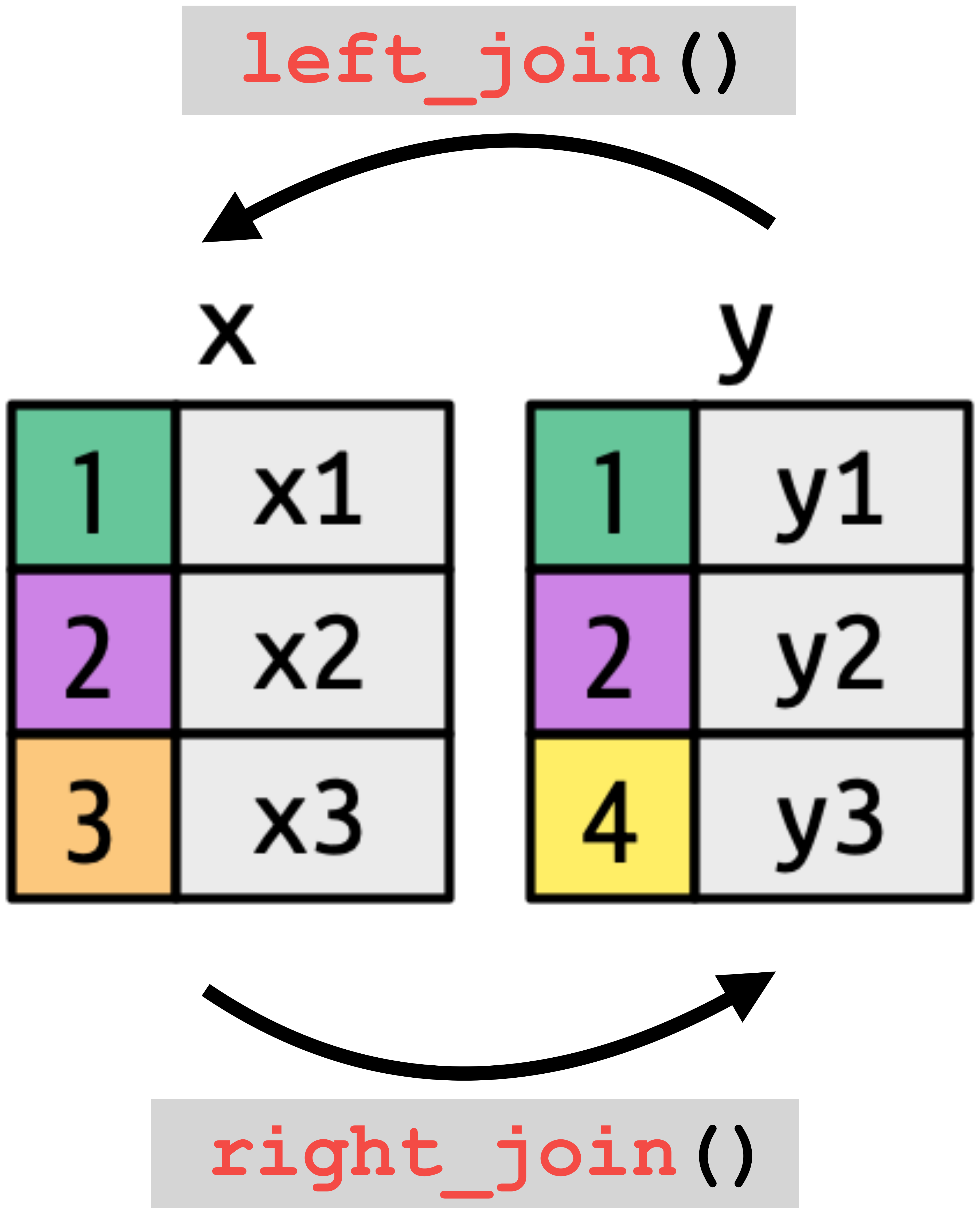
same variable across different datasets

R for Data Science

`tidyverse` package

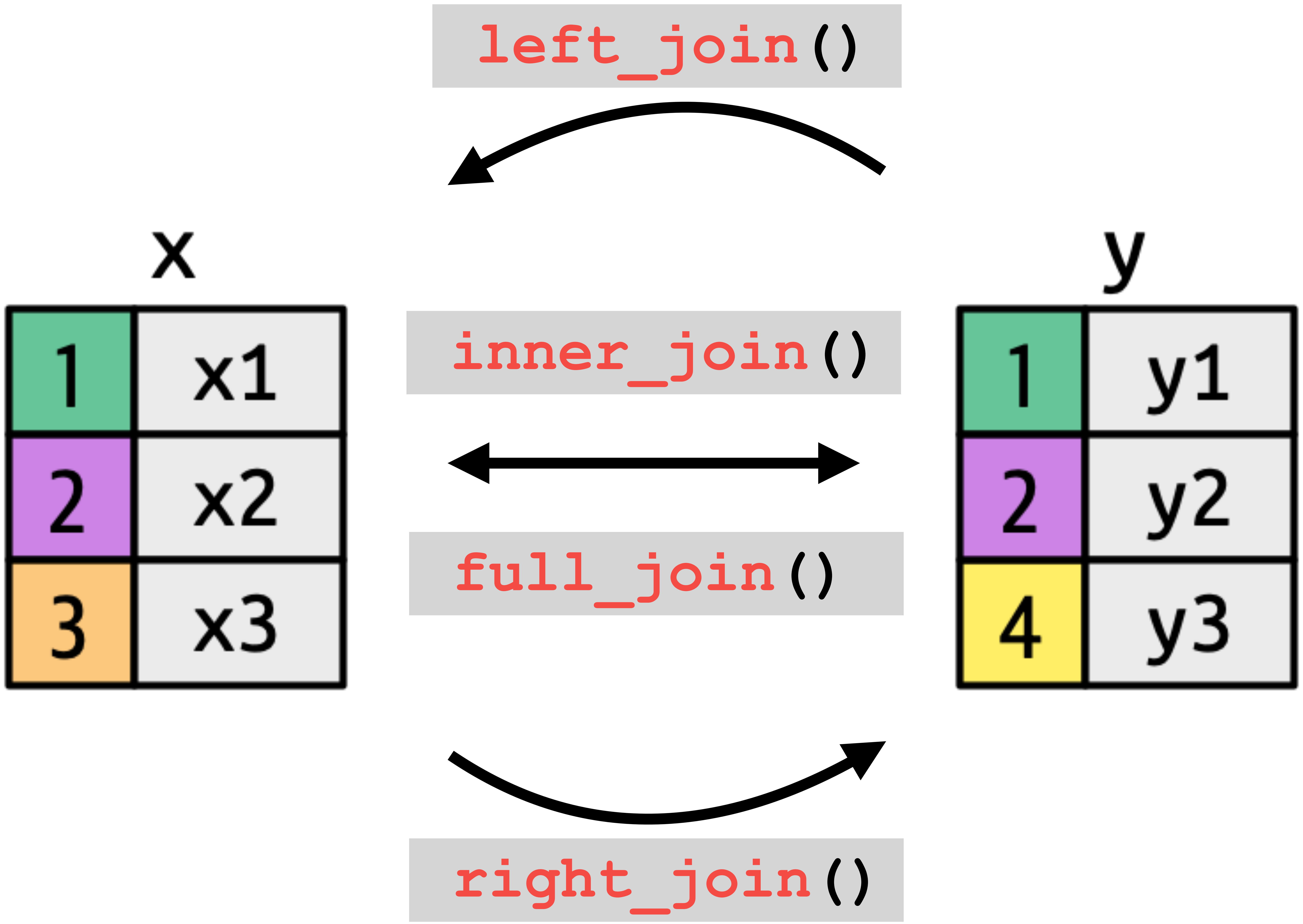
for manipulating data

Many different join options in `dplyr`



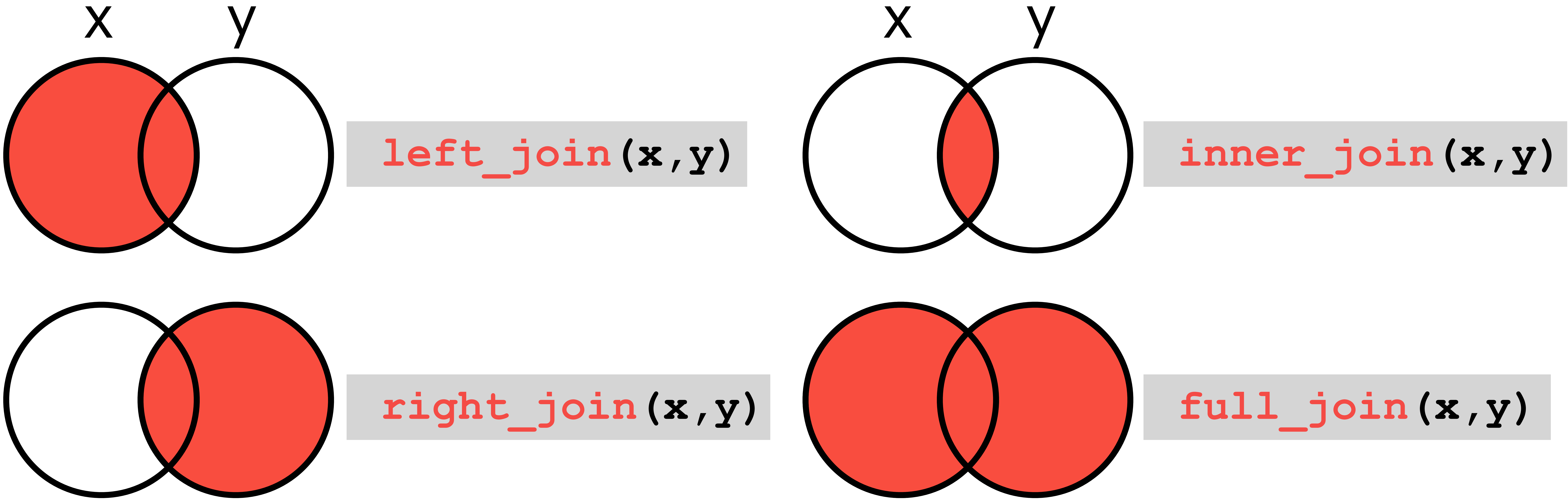
[R for Data Science]

Many different join options in `dplyr`

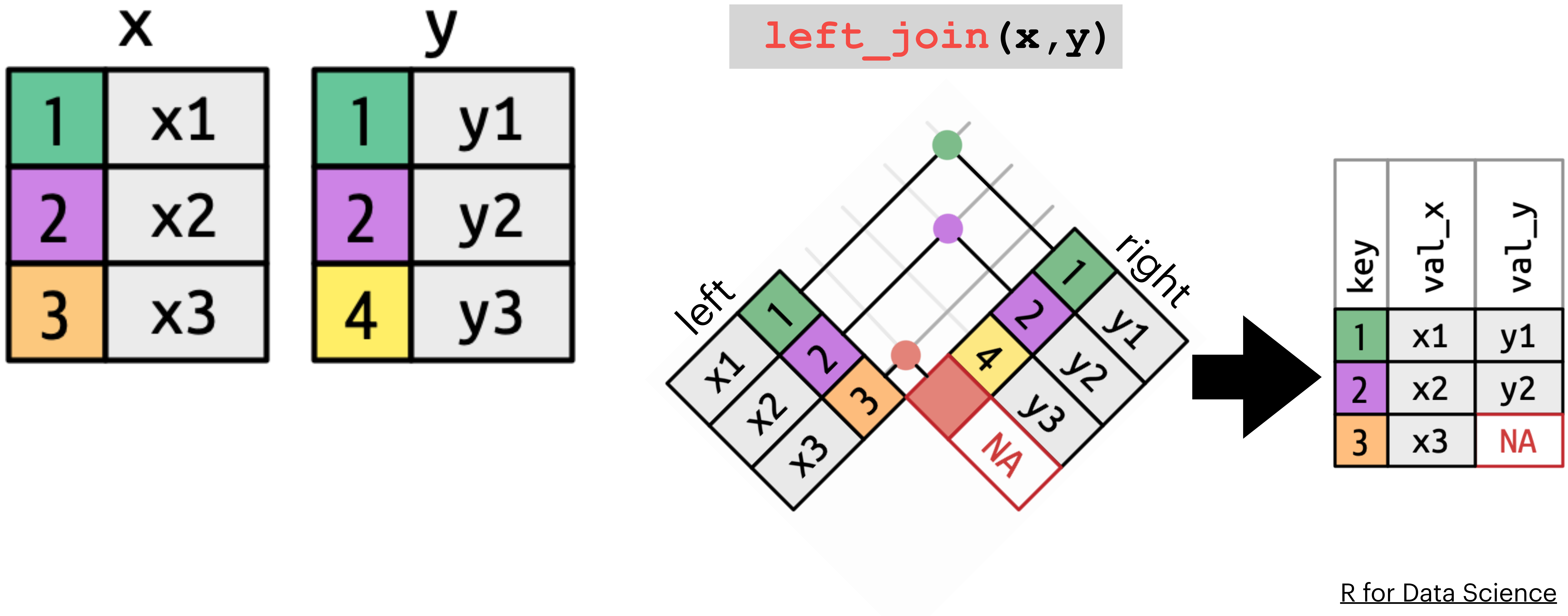


R for Data Science

Many different join options in `dplyr`



What do we get if we join these datasets?



[R for Data Science](#)

What do we get if we join these datasets?

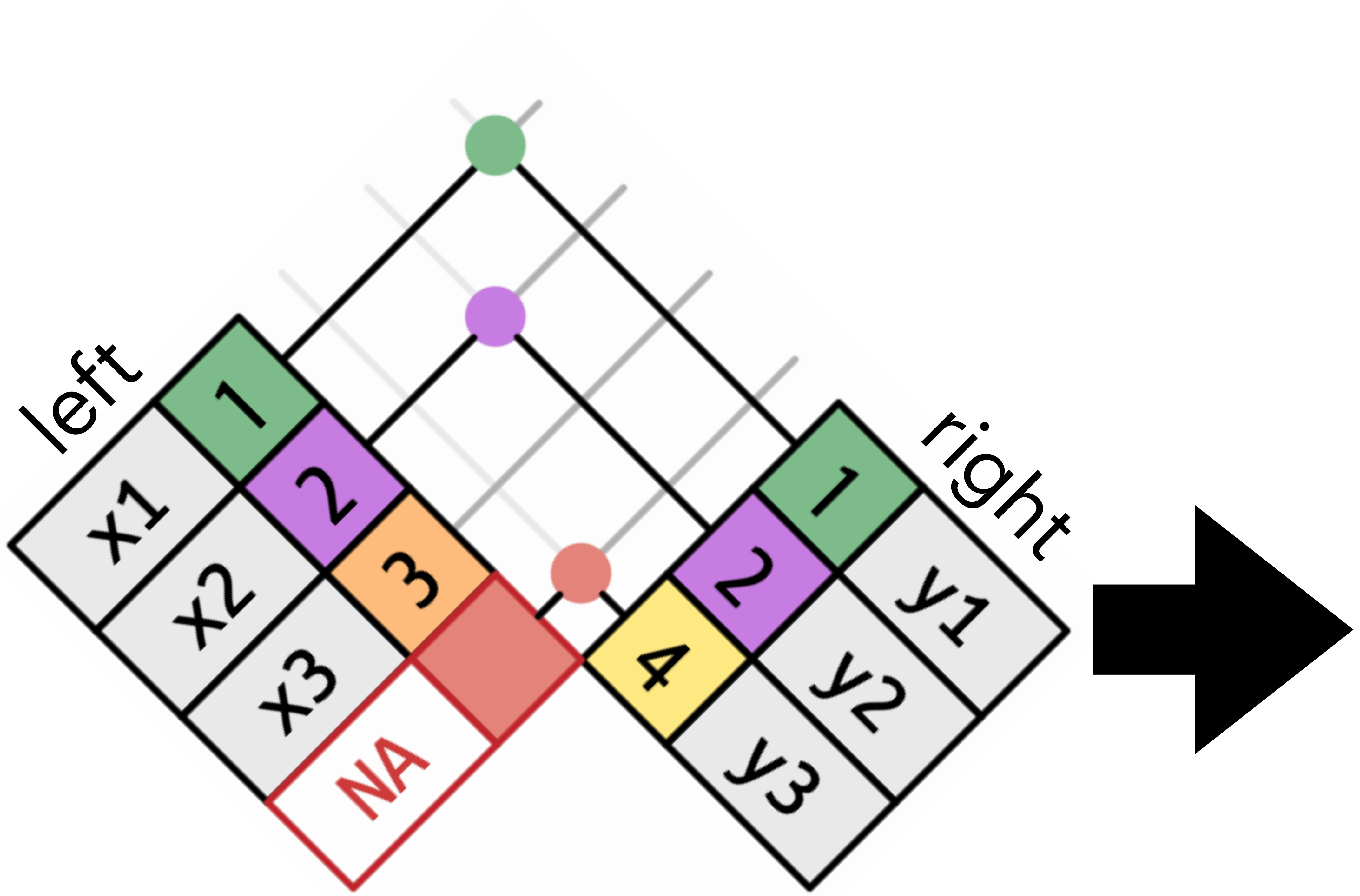
x

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3

```
right_join(x,y)
```



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

[R for Data Science]

What do we get if we join these datasets?

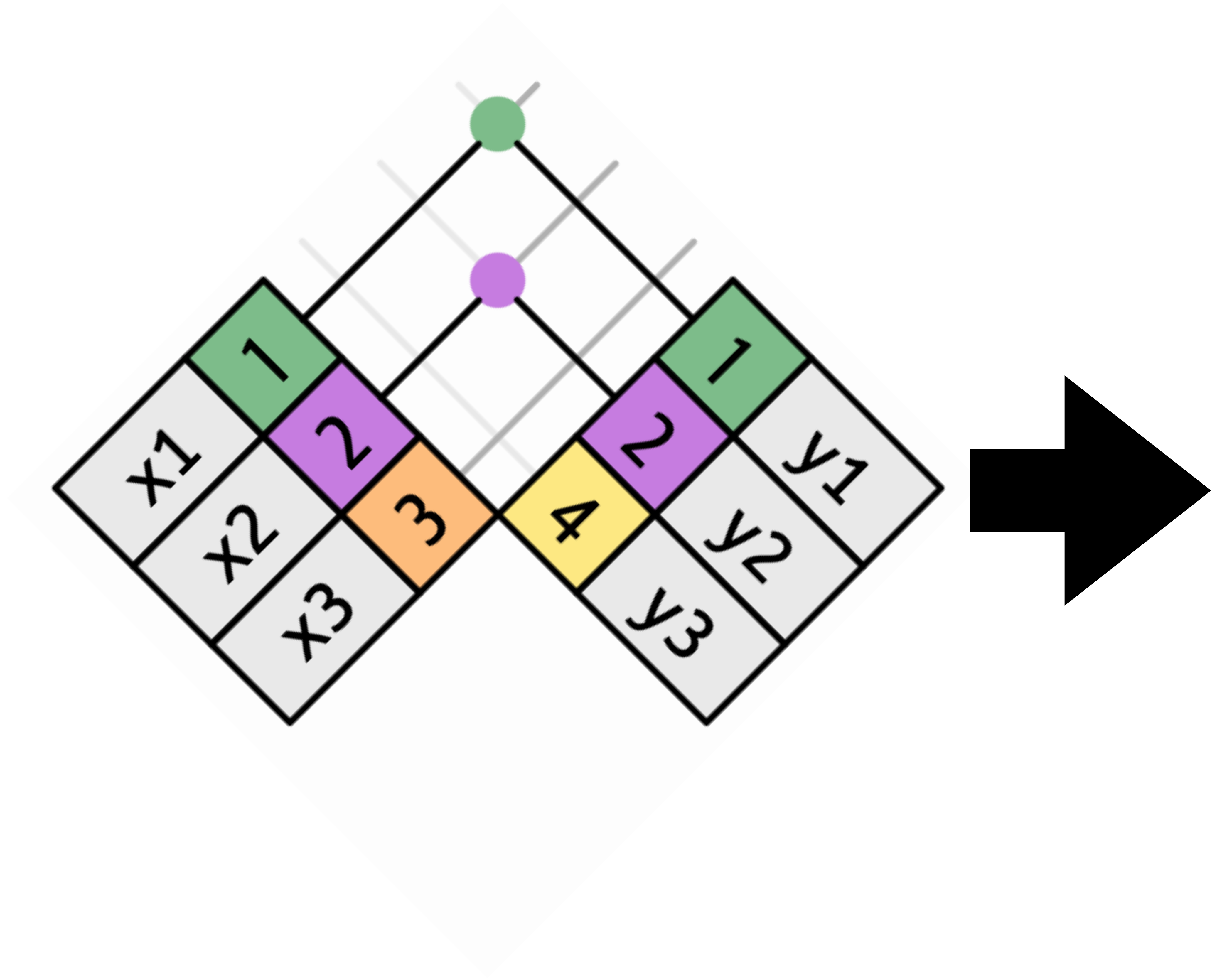
x

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3

`inner_join(x,y)`



key	val_x	val_y
1	x1	y1
2	x2	y2

R for Data Science

What do we get if we join these datasets?

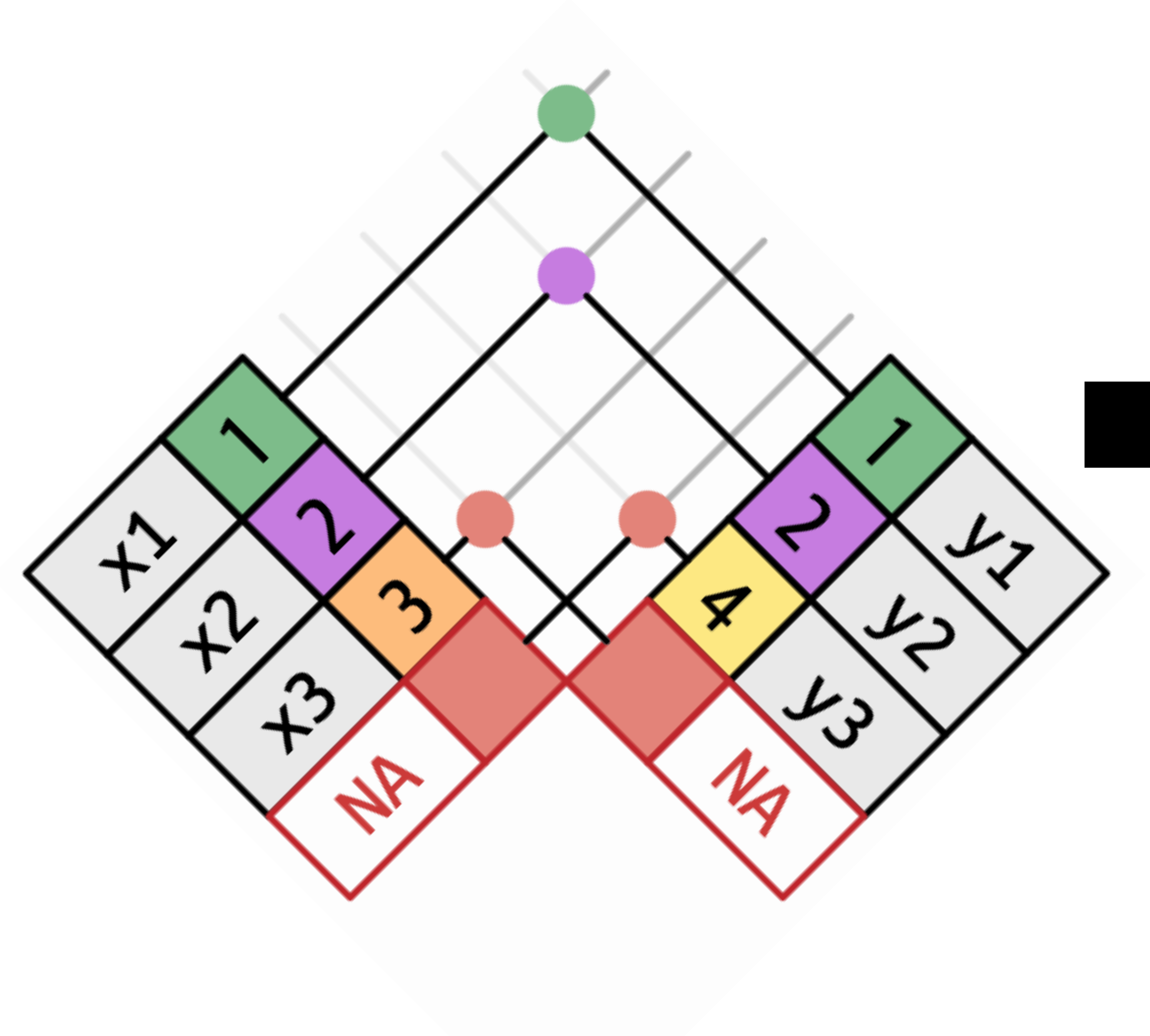
x

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3

```
full_join(x,y)
```

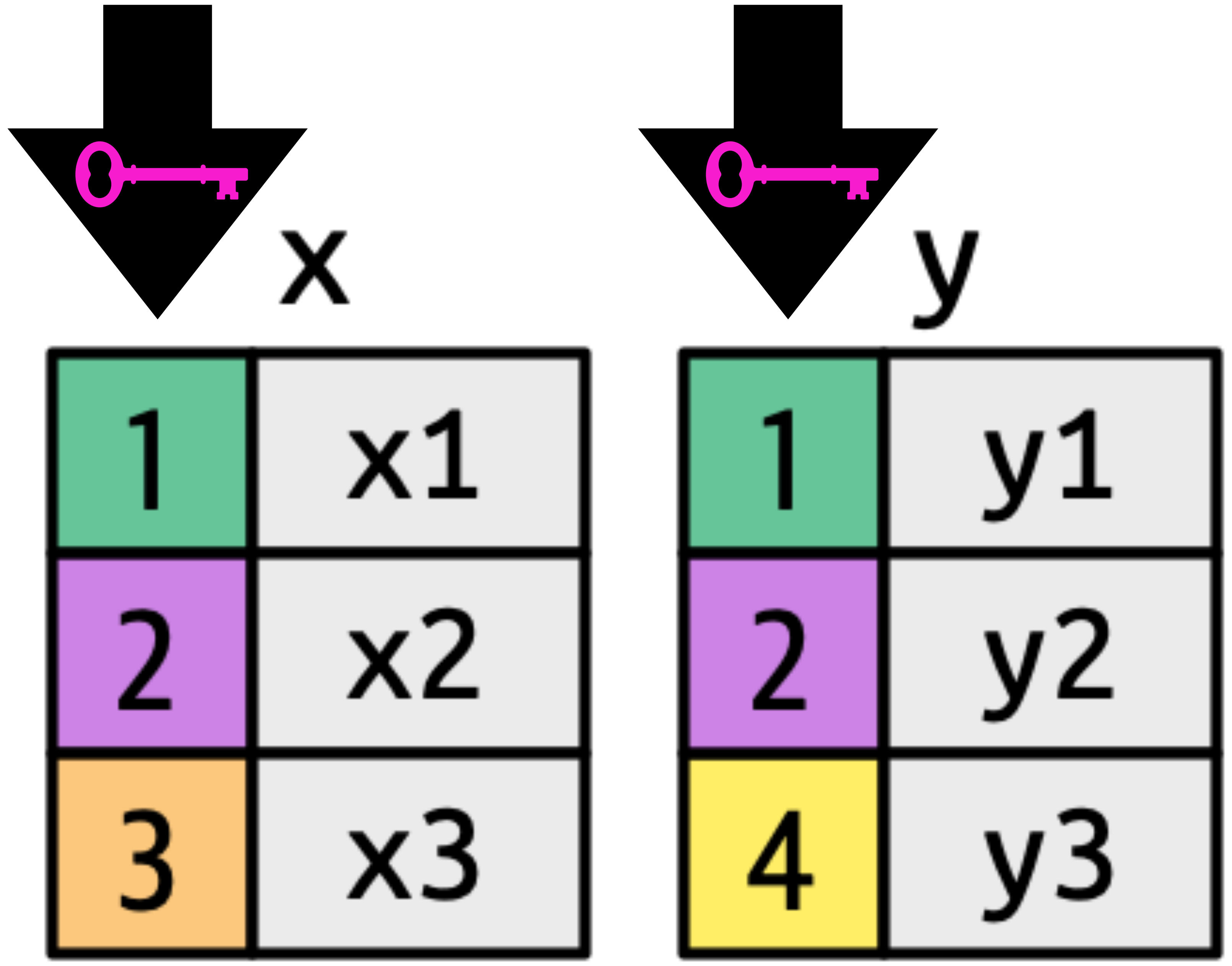


key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

R for Data Science

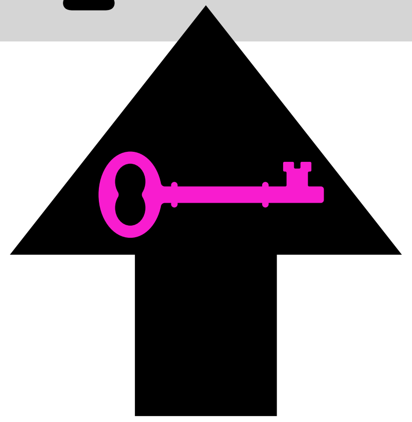
These functions use a “key”

Key = the column shared between the datasets



You don’t need to specify the key if the columns have the same names, otherwise…

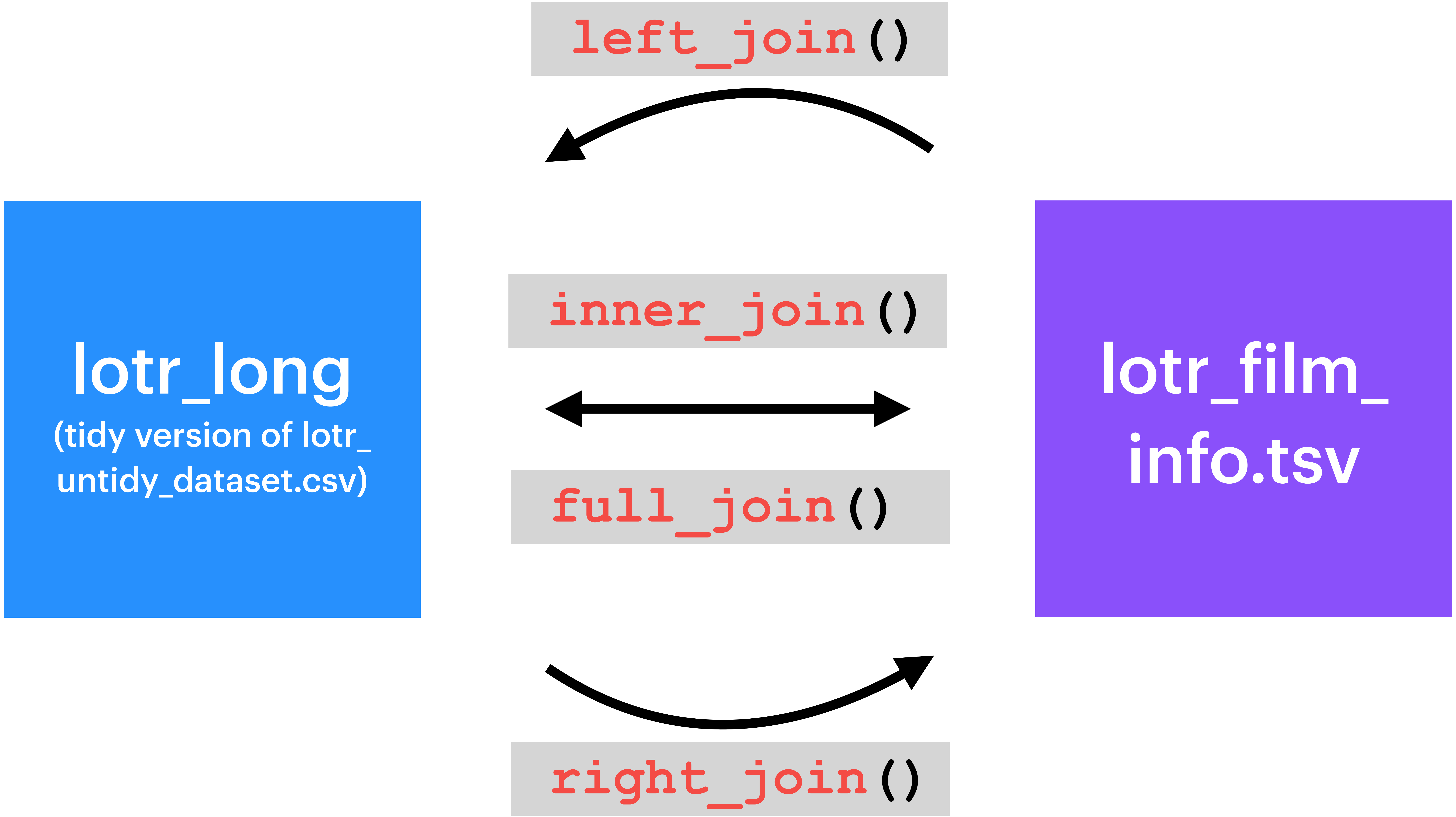
```
<TYPE>_join(x, y, by = )
```



R for Data Science

Practice, practice, practice…

Let’s practice joining two datasets!



Further reading

- Wickham & Grolemund 2017, *R for Data Science*, Chapter 5
- Wickham 2014, Tidy Data, *Journal of Statistical Software*
- Data tidying with `tidyr` Cheatsheet