QTM 151

Lab 3 – dplyr (join datasets)

Umberto Mignozzetti
Summer

Recap

We learned:

- `qplot`: quick way to make ggplot graphs.
- `ggplotly`: transform ggplot objects into nice plotly viz.
- `plot_ly`: create nice plotly graphs.

Do you have any questions about any of these contents?

The quiz is online. If you have any questions, feel free to come to the office hours

Our GitHub page is: https://github.com/umbertomig/qtm151

Getting Started

Getting Started: loading packages

```
# Loading tidyverse
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyv

## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse_c
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Loading data

```
band ← tribble(
   ~name,     ~band,
  "Mick",  "Stones",
  "John", "Beatles",
  "Paul", "Beatles")

instrument ← tribble(
    ~name,   ~plays,
   "John", "guitar",
   "Paul",   "bass",
  "Keith", "guitar")

instrument2 ← tribble(
    ~artist,   ~plays,
   "John", "guitar",
   "Paul",   "bass",
  "Keith", "guitar")
```

Loading data

```
# Loading PErisk dataset
PErisk ← read.csv('https://raw.githubusercontent.com/umbertomig/c
head(PErisk, 2)
```

```
##     country courts      barb2 prsexp2 prscorr2    gdpw2
## 1 Argentina      0 -0.7207754       1         3  9.69017
## 2 Australia      1 -6.9077550       5         4 10.30484
```

```
# First dataset
dat1 ← PErisk %>%
  filter(country %in% PErisk$country[1:5]) %>%
  select(country, courts:prsexp2)
dat1
```

```
##      country courts      barb2 prsexp2
## 1  Argentina      0 -0.7207754       1
## 2  Australia      1 -6.9077550       5
```

Join Datasets

Join Datasets

Join two or more datasets together is a common problem in data wrangling.

Lucky us, `dplyr` makes the job easy. Here are the functions we can use:

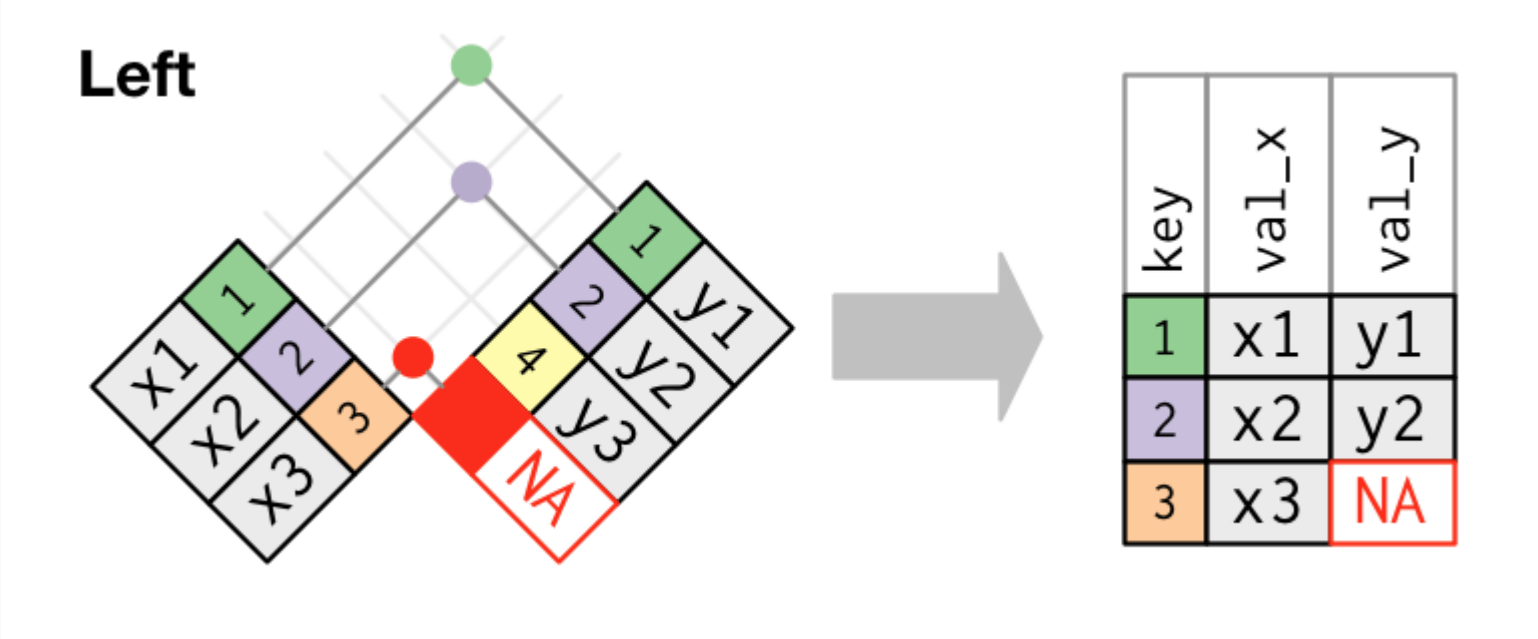
Function	Description
inner_join	Keep data in both datasets
left_join	Keep all data in the left dataset
right_join	Keep all data in the right dataset
full_join	Keep all data in both datasets
semi_join	Keep cases in the first that are also in the second
anti_join	Keep cases in the first that are NOT in the second

left_join

left_join

left_join(x, y, by="key variable"): join the datasets, keeping all the observations (rows) in x

- A key is a variable that uniquely identifies an observation, otherwise, we need multiple variables to identify an observation.



left_join

Example:

```
dat←left_join(band, instrument, by="name")
dat
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
```

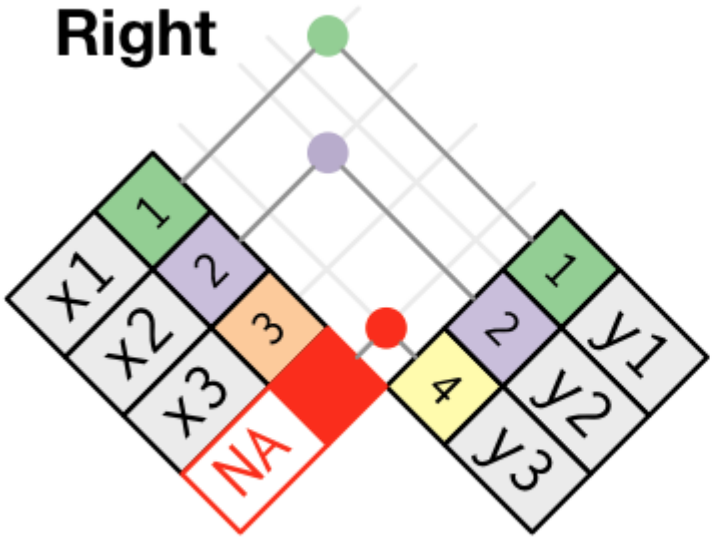
Your turn: Join the datasets `dat1` and `dat2` using left_join. Describe what happened.

right_join

right_join

right_join(x, y, by=""): keep all the observations (rows) in y

- The opposite way of *left_join()*



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

right_join

Example:

```
dat←right_join(band,instrument, by="name")
dat
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
## 3 Keith <NA>    guitar
```

Your turn: Join the datasets `dat1` and `dat2` using right_join. Describe what happened.

inner_join

inner_join

inner_join() keeps all the observations in **both** x and y

An inner join keeps observations that appear in both tables. But unmatched rows are not included in the result, it is easy to lose observations.

inner_join

Example:

```
inner_join(band, instrument, by="name")
```

```
## # A tibble: 2 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
```

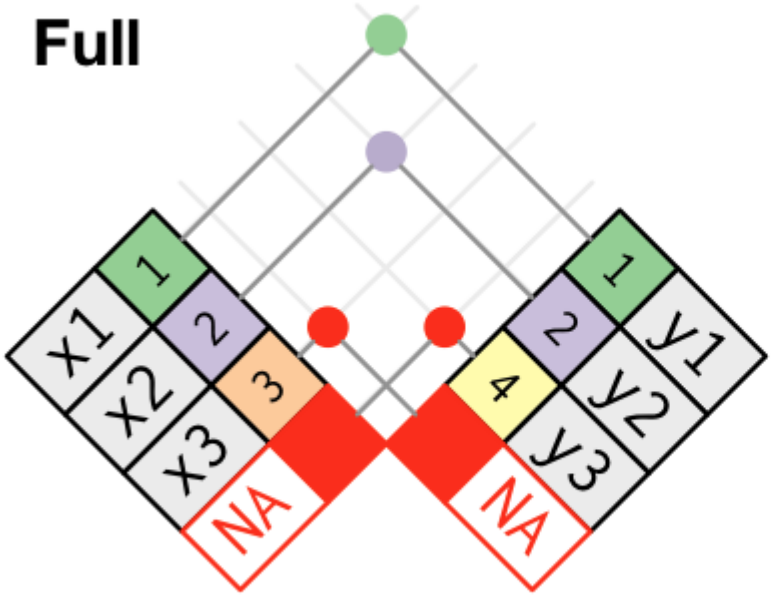
Your turn: Join the datasets `dat1` and `dat2` using inner_join. Describe what happened.

full_join

full_join

full_join keeps all observations in x and y

An full_join keeps observations that appear in either x or y.



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

full_join

Example:

```
full_join(band, instrument, by="name")
```

```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

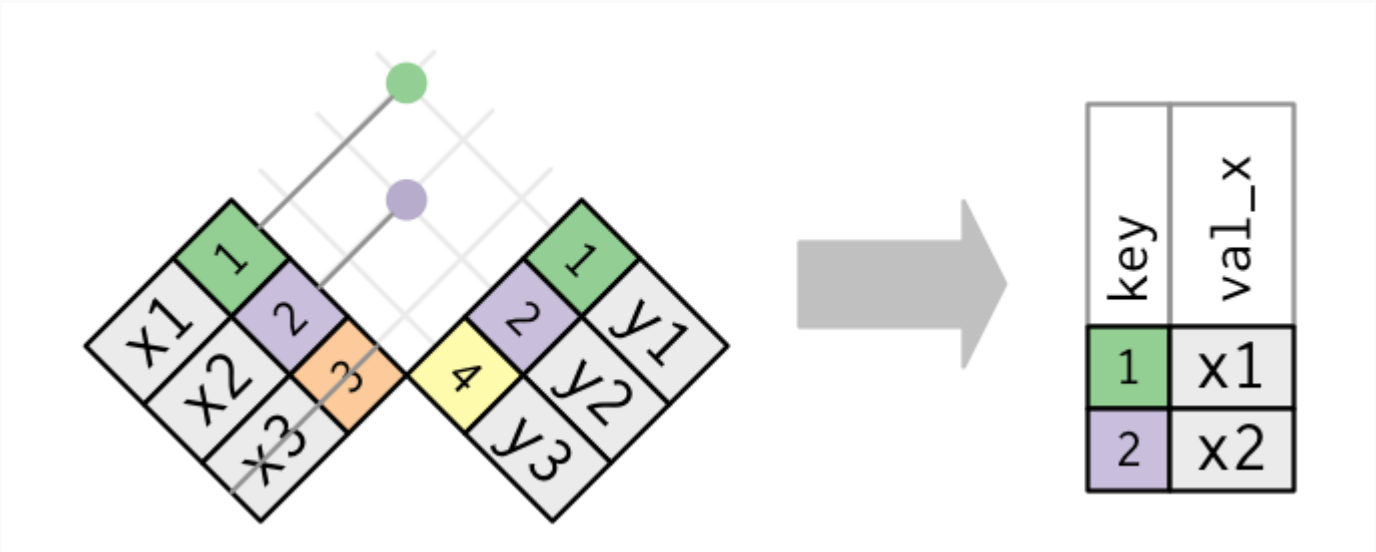
Your turn: Join the datasets `dat1` and `dat2` using full_join Describe what happened.

semi_join

semi_join

semi_join(x,y, by="") keeps all the observations of x that have a match in y

Use semi_join() to collect the artists in *band* that have instrument info in *instrument*.



semi_join

Example:

```
semi_join(band, instrument, by="name")
```

```
## # A tibble: 2 x 2
##   name  band
##   <chr> <chr>
## 1 John  Beatles
## 2 Paul  Beatles
```

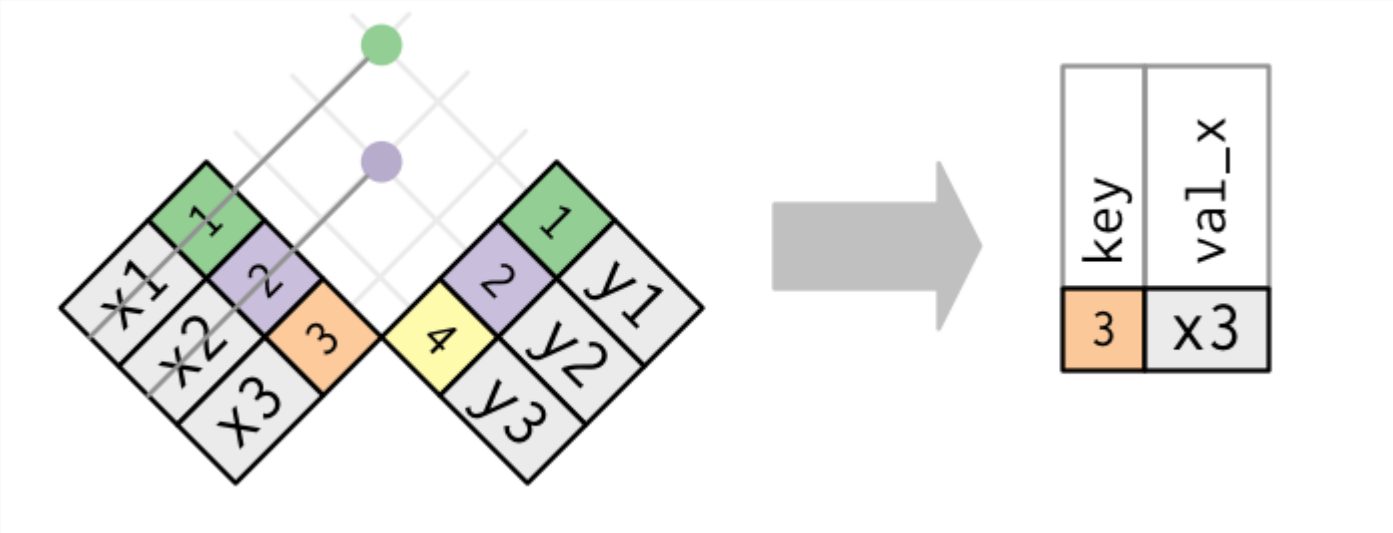
Your turn: Join the datasets `dat1` and `dat2` using semi_join. Describe what happened.

anti_join

anti_join

anti_join(x,y, by="") drops all the observations of x that have a match in y.

anti_join() also provide a great way to diagnose joins that go wrong.



anti_join

Example:

```
anti_join(band, instrument, by="name")
```

```
## # A tibble: 1 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
```

Your turn: Join the datasets `dat1` and `dat2` using anti_join. Describe what happened.

Questions?

See you next class!