13 Joining

Joins

- It’s rare that a data analysis involves only a single data frame.
- Typically you have many data frames, and you must join them together to answer the questions that you’re interested in.

Motivational example

Year 1

```
# A tibble: 7 × 4
     id gender ast101 ast102
  <int> <chr>   <dbl>  <dbl>
1   101 F          57     49
2   102 M          51     51
3   103 F          72     26
4   104 F          58     58
5   105 M          65     32
6   106 M          57     62
7   107 F          65     66
```

Year 2

```
# A tibble: 7 × 3
     id ast201 ast202
  <dbl>  <dbl>  <dbl>
1   101     77     43
2   102     72     34
3   103     65     41
4   104     76     39
5   105     75     37
6   106     70     35
7   201     76     65
```

Motivational example

- Create a variable grade which takes the following values:
  - 4.0 (for $\text{score} \geq 80$), 3.75 (for $75 \leq \text{score} < 80$),
  - 3.5 (for $70 \leq \text{score} < 75$), 3.25 (for $65 \leq \text{score} < 70$),
  - 3.0 (for $60 \leq \text{score} < 65$), , 2.5 (for $50 \leq \text{score} < 65$), and
  - 0 (for $\text{score} < 50$)

Motivational example

- Suppose, ast101 and ast201 are 4-credit course and other courses are of three credits
- Calculate the GPA for each student for two years separately
- Calculate CGPA, i.e., overall performance of each student
- Compare the performance of male and female on the basis of CGPA

Joins

- `dplyr` provides six join functions:
  - `left_join()`, `inner_join()`, `right_join()`, and `full_join()`
  - `semi_join()`, and `anti_join()`
- They all have the same interface:
  - they take a pair of data frames (`x` and `y`) and return a data frame

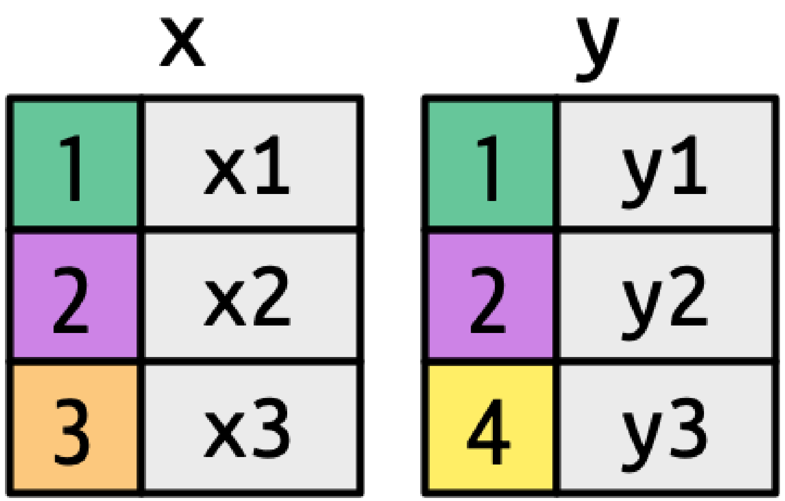
Mutating joins

- A mutating join allows you to combine variables from two data frames:
  - it first matches observations by their keys, then copies across variables from one data frame to the other
  - Like `mutate()`, the join functions add variables to the right
- There are four types of mutating join
  - `left_join()`, `inner_join()`, `right_join()`, `full_join()`

Joins

- Let’s define two simple tibbles x and y.

```
x <- tribble(
  ~key, ~val_x,
     1, "x1",
     2, "x2",
     3, "x3"
)
y <- tribble(
  ~key, ~val_y,
     1, "y1",
     2, "y2",
     4, "y3"
)
```



x

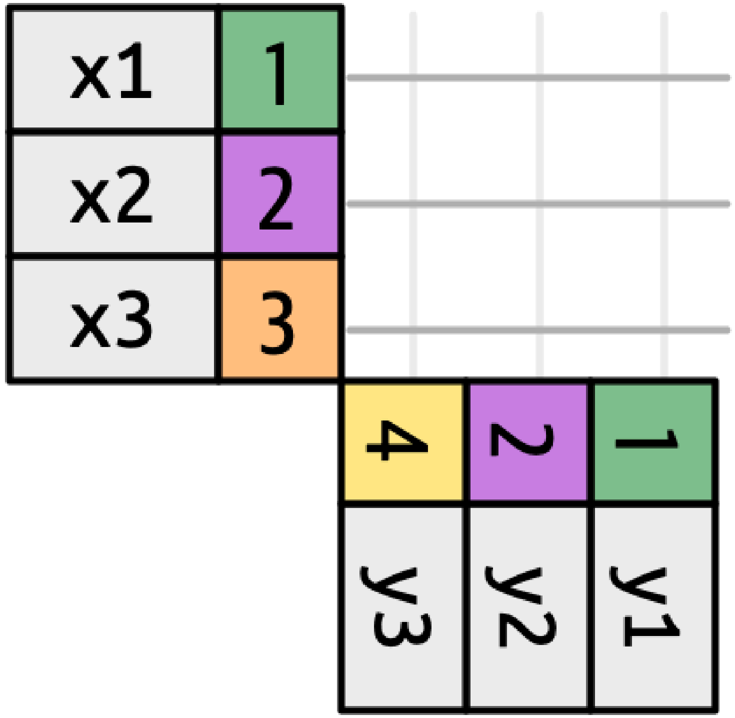
1	x1
2	x2
3	x3

y

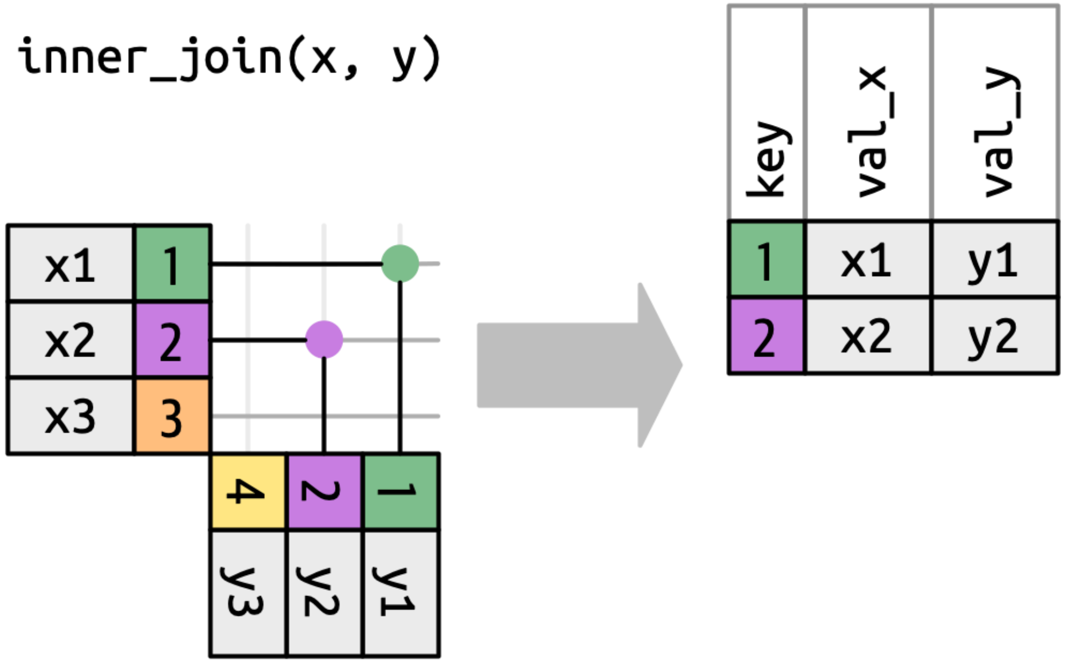
1	y1
2	y2
4	y3

Joins

- To understand how joins work, it’s useful to think of every possible match.
- Here we show that with a grid of connecting lines



Inner join

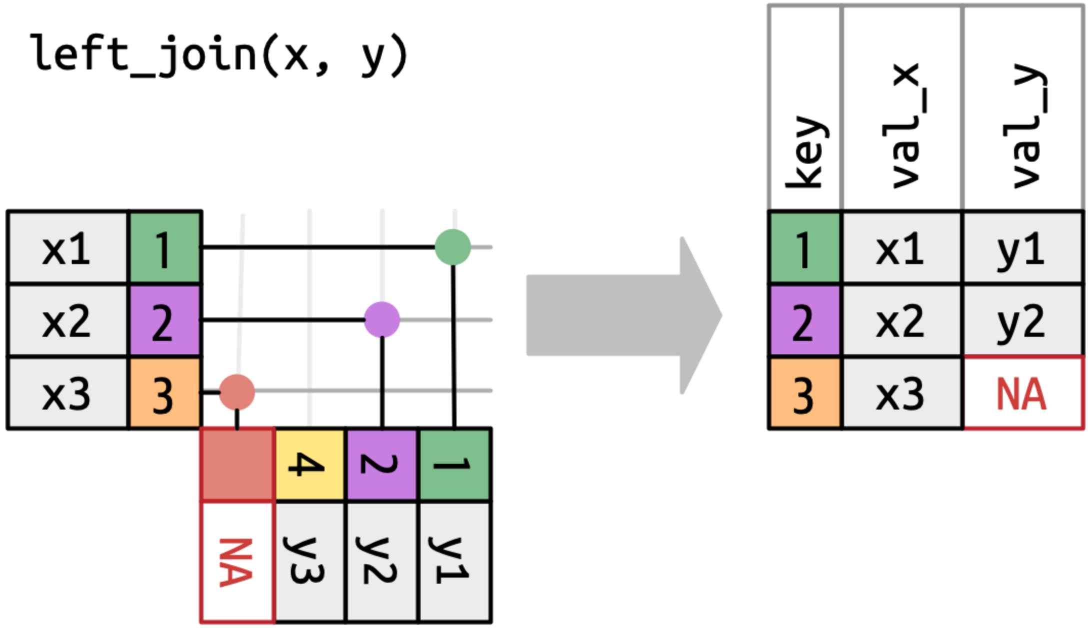


```
xy <- inner_join(x = x, y = y, by = "key")
```

```
xy
```

```
# A tibble: 2 × 3
    key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1   
2     2 x2    y2   
```

Inner join

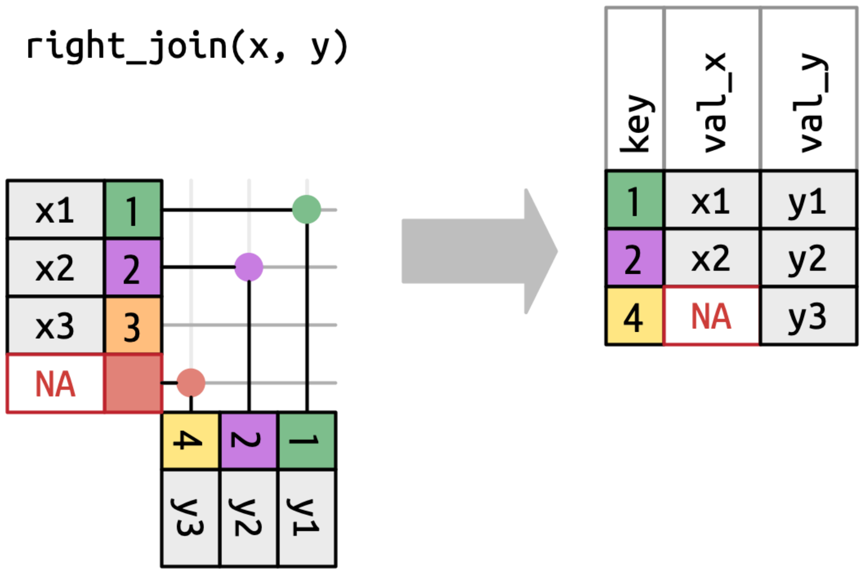


```
xy_left <- left_join(x = x, y = y, by = "key")
```

```
xy_left
```

```
# A tibble: 3 × 3
    key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1   
2     2 x2    y2   
3     3 x3    <NA> 
```

Inner join

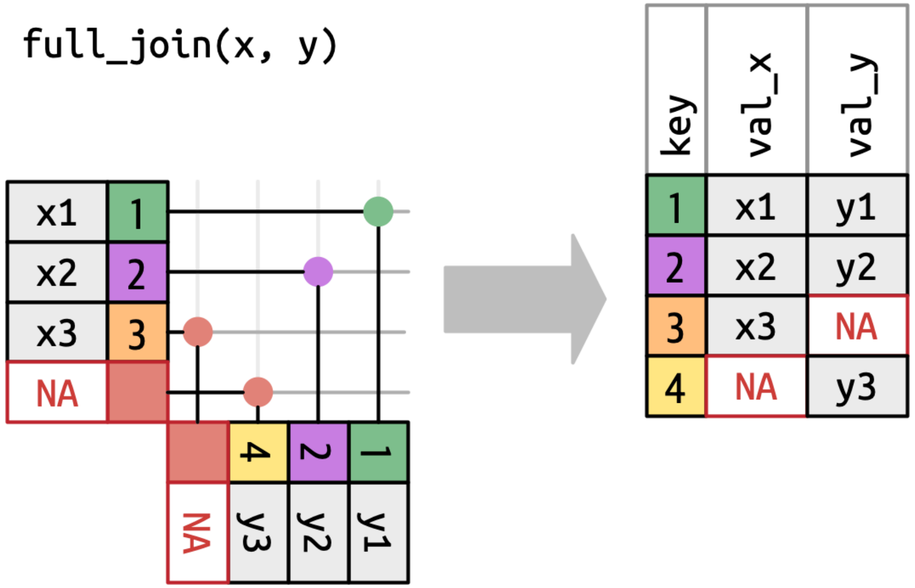


```
xy_right <- right_join(x = x, y = y,
                       by = "key")
```

```
xy_right
```

```
# A tibble: 3 × 3
    key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1
2     2 x2    y2
3     4 <NA>  y3
```

Inner join



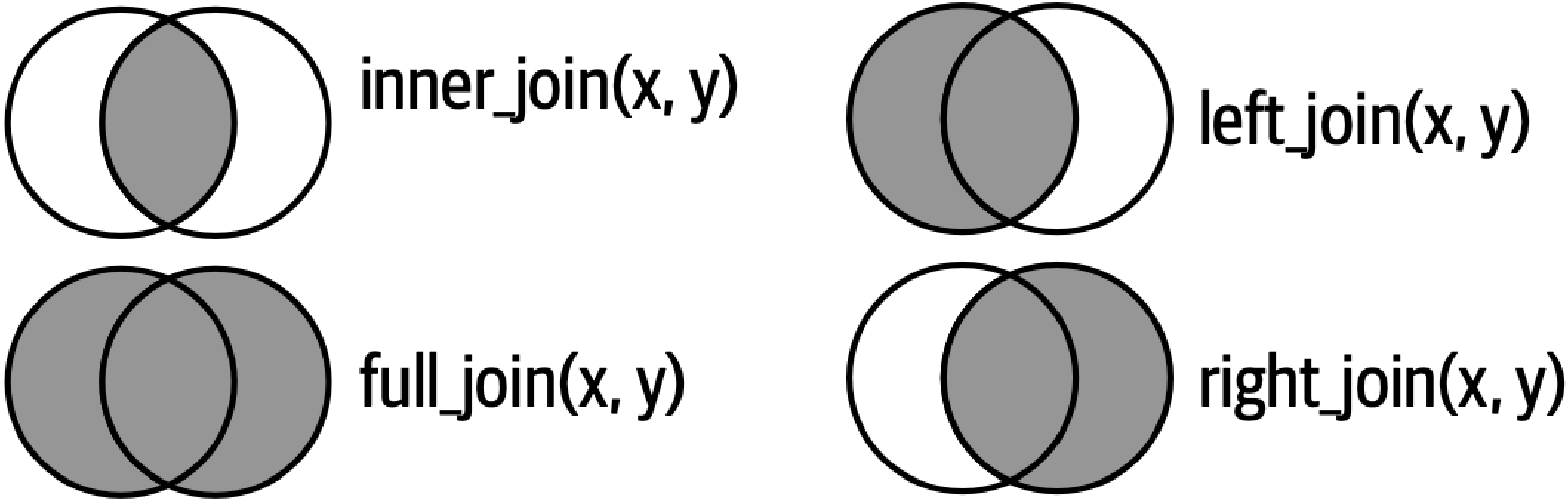
```
xy_full <- full_join(x = x, y = y, by = "key")
```

```
xy_full
```

```
# A tibble: 4 × 3
    key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1
2     2 x2    y2
3     3 x3    <NA>
4     4 <NA>  y3
```

Inner join

The following Venn diagrams showing the difference between inner, left, right, and full joins.



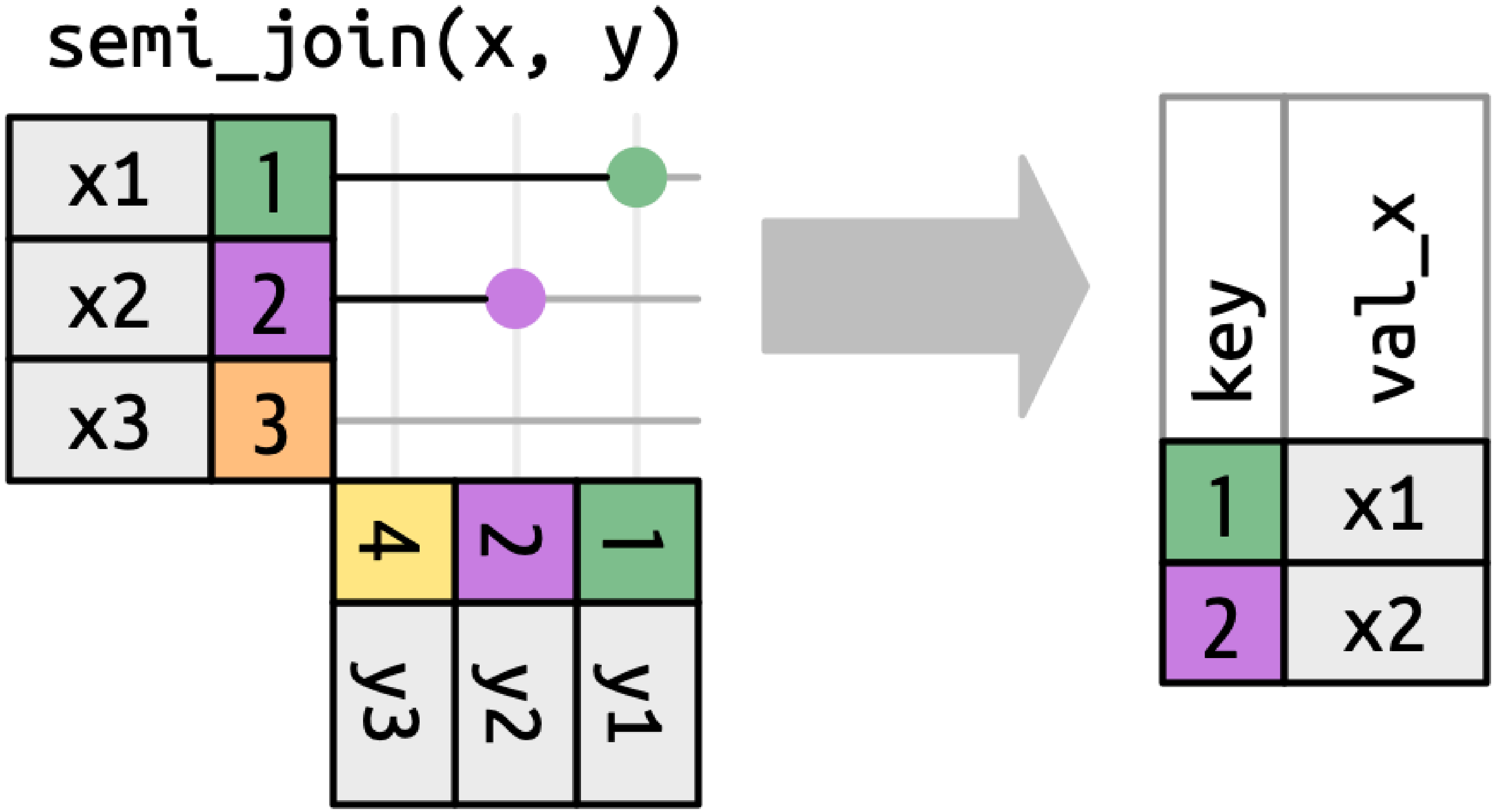
Filtering joins

Filtering joins

- Mutating joins add columns from y to x, matching rows based on the key.
- Filtering joins filter rows from x based on the presence or absence of matches in y.
- Two types of filtering join:
  - `semi_join()`, and `anti_join()`

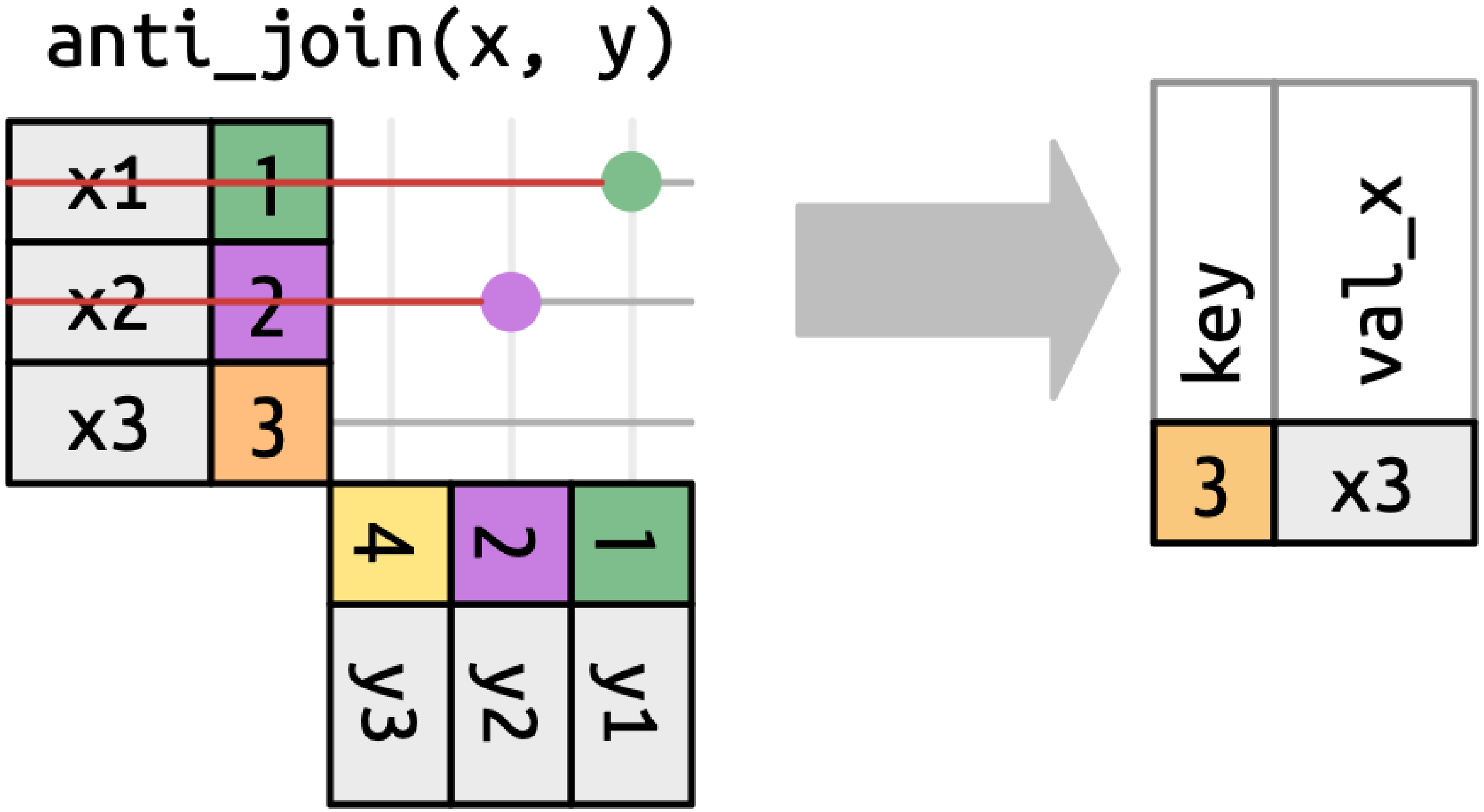
Filtering joins

- `semi_join()` return all rows from x with a match in y



Filtering joins

- `anti_join()` keeps rows in x that match zero rows in y



Exam data

Year 1

```
# A tibble: 7 × 4
     id gender ast101 ast102
  <int> <chr>   <dbl>  <dbl>
1   101 F          57     49
2   102 M          51     51
3   103 F          72     26
4   104 F          58     58
5   105 M          65     32
6   106 M          57     62
7   107 F          65     66
```

Year 2

```
# A tibble: 7 × 3
     id ast201 ast202
  <dbl>  <dbl>  <dbl>
1   101     77     43
2   102     72     34
3   103     65     41
4   104     76     39
5   105     75     37
6   106     70     35
7   201     76     65
```

- Calculate CGPA, i.e., overall performance of each student
- Compare the performance of male and female on the basis of CGPA