Relational data

Four main types of operations with two tables

- **Binding,** which simply stacks tables on top of or beside each other
- **Mutating joins**, which add new variables to one data frame from matching observations in another.
- **Filtering joins**, which filter observations from one data frame based on whether or not they match an observation in the other table.
- **Set operations**, which treat observations as if they were set elements.

Keys

- A variable **(or set of variables)** that uniquely identifies an observation

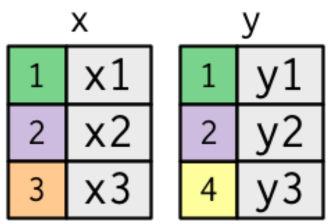
  - A **primary key** uniquely identifies an observation in its own table [can be a set of variables]. For example, planes$tailnum is a primary key because it uniquely identifies each plane in the planes table.

  - A **foreign key** uniquely identifies an observation in another table [can be a set of variables]. For example, the flights$tailnum is a foreign key because it appears in the flights table where it matches each flight to a unique plane.

Relations

- Typically one-to-many
  - Each flight has one plane, but each plane has many flights
- Can also be many-to-many
  - Each airline flies to many airports; each airport hosts many airlines
- Can also be one-to-one

Understanding joins



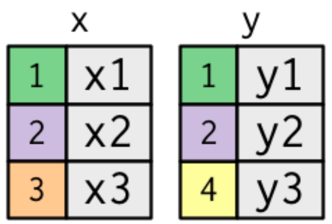
x

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3

Understanding joins



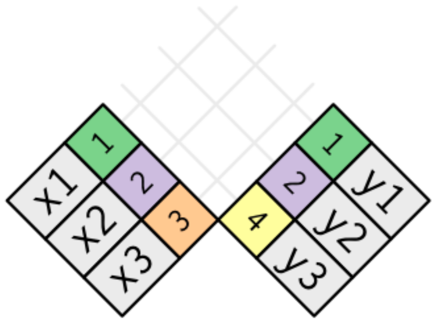
x

1	x1
2	x2
3	x3

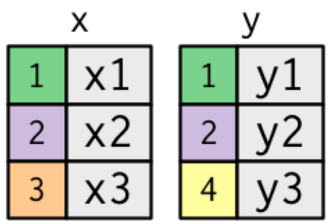
y

1	y1
2	y2
4	y3

Each potential match



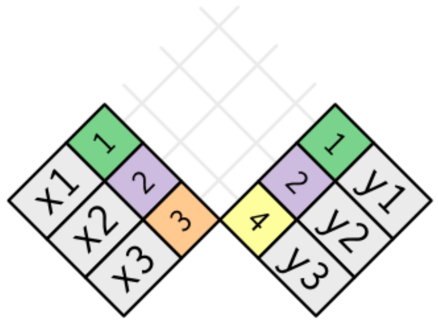
Understanding joins



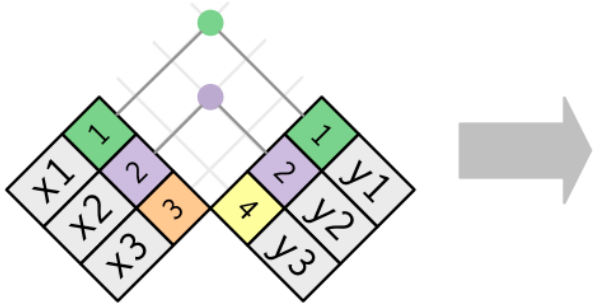
x	
1	x1
2	x2
3	x3

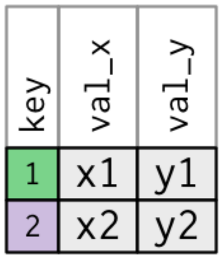
y	
1	y1
2	y2
4	y3

Each potential match



Number of actual matches



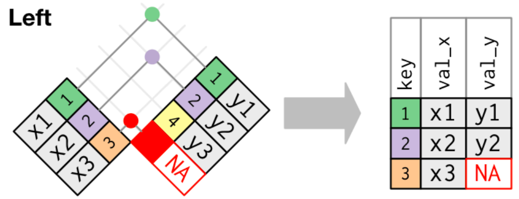


key	val_x	val_y
1	x1	y1
2	x2	y2

Inner join: Unmatched rows are not included in the output

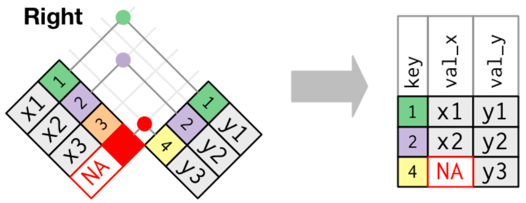
Outer joins

Keeps all observations in x



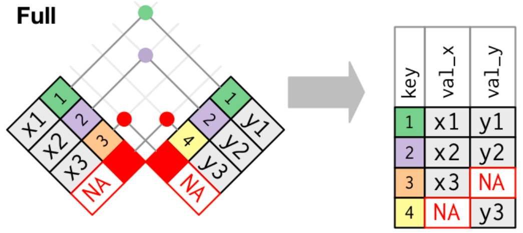
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Keeps all observations in y



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

Keeps all observations in x and y



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

Combine Data Sets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

dplyr::**left_join(a, b, by = "x1")**

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

dplyr::**right_join(a, b, by = "x1")**

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

dplyr::**inner_join(a, b, by = "x1")**

Join data. Retain only rows in both sets.

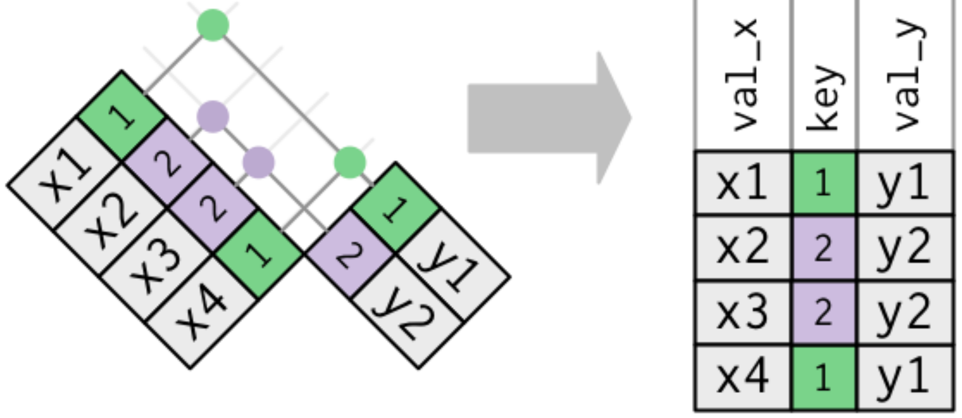
x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

dplyr::**full_join(a, b, by = "x1")**

Join data. Retain all values, all rows.

Duplicate keys

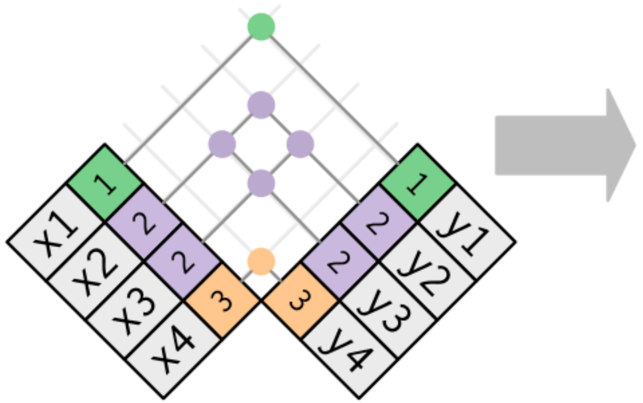
One table has duplicate keys (typically a one-to-many relationship)
e.g. “dest” in the flights tibble



val_x	key	val_y
x1	1	y1
x2	2	y2
x3	2	y2
x4	1	y1

Duplicate keys

Both tables have duplicate keys (typically an error)

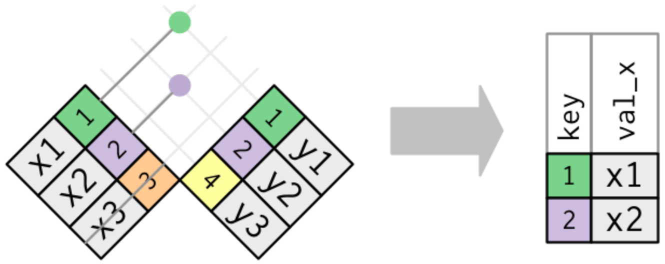


key	val1_x	val1_y
1	x1	y1
2	x2	y2
2	x2	y3
2	x3	y2
2	x3	y3
3	x4	y4

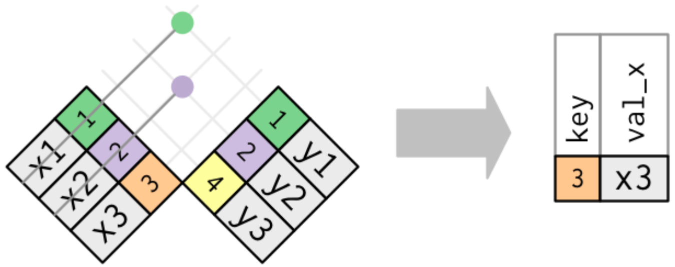
Filtering

- semi_join(x, y) **keeps** all observations in x that have a match in y.
- anti_join(x, y) **drops** all observations in x that have a match in y.

Semi-join



Anti-join



Exercises

- Filter flights to only show flights with planes that have flown at least 100 flights
- Combine fueleconomy::vehicles and fueleconomy::common to find only the records for the most common models

Join problems – how to troubleshoot

- Start by identifying the variables that form the primary key in each table based on your understanding of the data

- Check that none of the variables in the primary key are missing. If a value is missing then it can’t identify an observation!

- Check that your foreign keys match primary keys in another table. The best way to do this is with an anti_join()

Tibbles

- “Opinionated data.frames”
  - Never changes the type of the inputs (e.g. it never converts strings to factors!)
  - Never changes the names of variables
  - Never creates row names
  - Never partial matching to variable names

Printing

- tibbles vs. data.frames

Subsetting

- With tibbles: primarily with dplyr::filter() and dplyr::select()
  - ‘[’ always returns a tibble
- In data.frames: primarily with ‘[’
  - ‘[’ an either return a data.frame or a vector

Data classes in R

Exercises

1. How can you tell if an object is a tibble? (Hint: try printing `mtcars`, which is a regular data frame).

2. Compare and contrast the following operations on a `data.frame` and equivalent tibble. What is different? Why might the default data frame behaviours cause you frustration?

```r
df <- data.frame(abc = 1, xyz = "a")
df$x
df[, "xyz"]
df[, c("abc", "xyz")]
```

3. If you have the name of a variable stored in an object, e.g. `var <- "mpg"`, how can you extract the reference variable from a tibble?

4. Practice referring to non-syntactic names in the following data frame by:

    1. Extracting the variable called `1`.

    2. Plotting a scatterplot of `1` vs `2`.

    3. Creating a new column called `3` which is `2` divided by `1`.

    4. Renaming the columns to `one`, `two` and `three`.

```
annoying <- tibble(
  `1` = 1:10,
  `2` = `1` * 2 + rnorm(length(`1`))
)
```

Exercises

- Explore the distribution of rincome (reported income). What makes the default bar chart hard to understand? How could you improve the plot?

- What is the most common relig in this survey? What's the most common partyid?

- Now create a similar plot looking at how average age varies across reported income level

Exercises

- Add the location of the origin *and* destination (i.e. the lat and lon) to flights.

- IF MORE TIME:
  - Is there a relationship between the age of a plane and its delays?