Day 10: Relational Data

Erin Rossiter

03 April, 2023

Announcements

Announcements

- project update feedback in Slack
  - » if you want additional feedback, come to office hours
- PS08 due tomorrow
- No class on Monday because of university holiday
  - » No office hours either!
  - » **Make up office hours Wednesday 4/12 10:30am-12:30pm**

Announcements

- project update feedback in Slack
  - if you want additional feedback, come to office hours
- PS08 due tomorrow
- No class on Monday because of university holiday
  - No office hours either!
  - **Make up office hours Wednesday 4/12 10:30am-12:30pm**

Announcements

- project update feedback in Slack
  - » if you want additional feedback, come to office hours
- PS08 due tomorrow
- No class on Monday because of university holiday
  - » No office hours either!
  - » **Make up office hours Wednesday 4/12 10:30am-12:30pm**

Announcements

- project update feedback in Slack
  - if you want additional feedback, come to office hours
- PS08 due tomorrow
- No class on Monday because of university holiday
  - No office hours either!
  - **Make up office hours Wednesday 4/12 10:30am-12:30pm**

Announcements

- project update feedback in Slack
  - if you want additional feedback, come to office hours
- PS08 due tomorrow
- No class on Monday because of university holiday
  - No office hours either!
  - **Make up office hours Wednesday 4/12 10:30am-12:30pm**

Announcements

- project update feedback in Slack
  - » if you want additional feedback, come to office hours
- PS08 due tomorrow
- No class on Monday because of university holiday
  - » No office hours either!
  - » **Make up office hours Wednesday 4/12 10:30am-12:30pm**

Today: relational data

Today: relational data

- Most complex analyses involve more than one table of data
- Certainly most active databases are not just rectangles!
- Modern databases are **relational**, where we know how rows in each rectangle are related to each other
- We can learn how to work cleanly with such data

Today: relational data

- Most complex analyses involve more than one table of data
- Certainly most active databases are not just rectangles!
- Modern databases are **relational**, where we know how rows in each rectangle are related to each other
- We can learn how to work cleanly with such data

Today: relational data

- Most complex analyses involve more than one table of data
- Certainly most active databases are not just rectangles!
- Modern databases are **relational**, where we know how rows in each rectangle are related to each other
- We can learn how to work cleanly with such data

Today: relational data

- Most complex analyses involve more than one table of data
- Certainly most active databases are not just rectangles!
- Modern databases are **relational**, where we know how rows in each rectangle are related to each other
- We can learn how to work cleanly with such data

Today: relational data

- Most complex analyses involve more than one table of data
- Certainly most active databases are not just rectangles!
- Modern databases are **relational**, where we know how rows in each rectangle are related to each other
- We can learn how to work cleanly with such data

What is relational data?

- A database in which information is stored in a number of database tables
- Each database table has its own rows and columns
- **The different tables in a database can be related**
- Having data in separate but related tables is:
  - an efficient way to store
  - and efficient way to retrieve

What is relational data?

- A database in which information is stored in a number of database tables
- Each database table has its own rows and columns
- **The different tables in a database can be related**
- Having data in separate but related tables is:
  - an efficient way to store
  - and efficient way to retrieve

What is relational data?

- A database in which information is stored in a number of database tables
- Each database table has its own rows and columns
- **The different tables in a database can be related**
- Having data in separate but related tables is:
  - an efficient way to store
  - and efficient way to retrieve

What is relational data?

- A database in which information is stored in a number of database tables
- Each database table has its own rows and columns
- **The different tables in a database can be related**
- Having data in separate but related tables is:
  - an efficient way to store
  - and efficient way to retrieve

What is relational data?

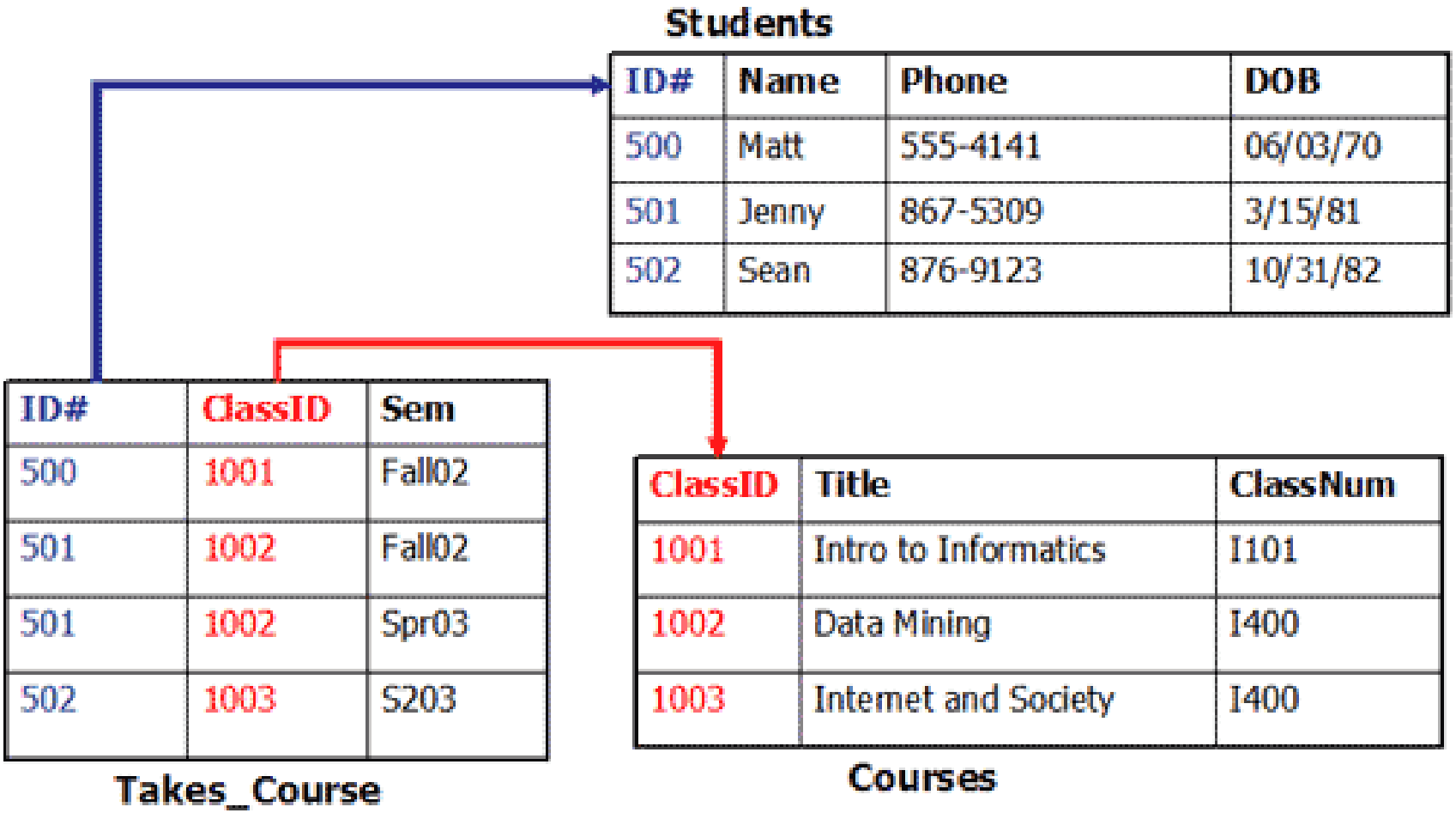
- A database in which information is stored in a number of database tables
- Each database table has its own rows and columns
- **The different tables in a database can be related**
- Having data in separate but related tables is:
  - an efficient way to store
  - and efficient way to retrieve

What is relational data?

- A database in which information is stored in a number of database tables
- Each database table has its own rows and columns
- **The different tables in a database can be related**
- Having data in separate but related tables is:
  - an efficient way to store
  - and efficient way to retrieve

Example

Example



Students

ID#	Name	Phone	DOB
500	Matt	555-4141	06/03/70
501	Jenny	867-5309	3/15/81
502	Sean	876-9123	10/31/82

ID#	ClassID	Sem
500	1001	Fall02
501	1002	Fall02
501	1002	Spr03
502	1003	S203

Takes_Course

ClassID	Title	ClassNum
1001	Intro to Informatics	I101
1002	Data Mining	I400
1003	Internet and Society	I400

Courses

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday. . .)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday...)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - space reasons
  - sharing reasons (maybe you make a GUI someday...)
  - network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - Simple example: Supreme Court Database
  - Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday. . .)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday. . .)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday. . .)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday...)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Why are we learning this?

- You might want to store a database you collect relationally
  - » space reasons
  - » sharing reasons (maybe you make a GUI someday. . .)
  - » network data is inherently like this (nodes and edges)
- Or, you might need to access data stored this way
  - » Simple example: Supreme Court Database
  - » Any examples?
- Or just for data cleaning!

Running example

- Mayor-level information
- Tweet-level information

Two topics today

There are various ways we might want to work across levels

- *Joins*, where you make a new variables constructed form matched observations in the other
  - » Ex: We have mayor info and want to add Twitter info to that dataset
- *Creating/storing/querying* datasets fully with SQL
  - » Exposure to this today

Two topics today

There are various ways we might want to work across levels

- *Joins*, where you make a new variables constructed form matched observations in the other
  - Ex: We have mayor info and want to add Twitter info to that dataset
- *Creating/storing/querying* datasets fully with SQL
  - Exposure to this today

Two topics today

There are various ways we might want to work across levels

- *Joins*, where you make a new variables constructed form matched observations in the other
  - » Ex: We have mayor info and want to add Twitter info to that dataset
- *Creating/storing/querying* datasets fully with SQL
  - » Exposure to this today

Two topics today

There are various ways we might want to work across levels

- *Joins*, where you make a new variables constructed form matched observations in the other
  - Ex: We have mayor info and want to add Twitter info to that dataset
- *Creating/storing/querying* datasets fully with SQL
  - Exposure to this today

Two topics today

There are various ways we might want to work across levels

- *Joins*, where you make a new variables constructed form matched observations in the other
  - Ex: We have mayor info and want to add Twitter info to that dataset
- *Creating/storing/querying* datasets fully with SQL
  - Exposure to this today

Joins

Joins

You'll do lots of joins in your life!

- explicitly relational data or not
- examples?

Joins

You'll do lots of joins in your life!

- explicitly relational data or not
- examples?

Joins

You'll do lots of joins in your life!

- explicitly relational data or not
- examples?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - » Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - » Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - » Variable(s) to join by, specified with `by`
  - » Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - Variable(s) to join by, specified with `by`
  - Allow multiple matches or not?

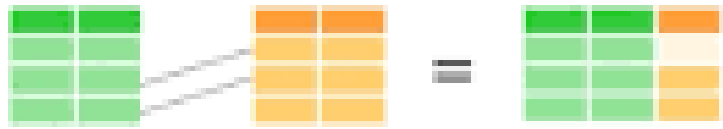
Kinds of joins and dplyr syntax

Consider we have dataframes `x` and `y`

- 4 most common types of joins:
  - » Inner join
    - `inner_join()` keeps observations from `x` that have a matching key in `y`
  - » Outer joins keep observations that appear in at least 1 dataframe:
    - `left_join()` keeps all observations in `x`
    - `right_join()` keeps all observations in `y`
    - `full_join()` keeps all observations in `x` and `y`
- Also need to know:
  - » Variable(s) to join by, specified with `by`
  - » Allow multiple matches or not?

Joins

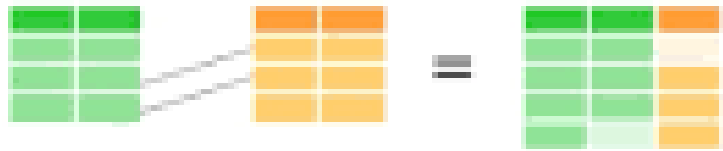
`left_join()`



`right_join()`

`inner_join()`

`full_join()`



Working with remote data

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - » Your data is already stored this way
  - » You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - » It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - » Your data is already stored this way
  - » You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - » It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - » Your data is already stored this way
  - » You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - » It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - Your data is already stored this way
  - You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - » Your data is already stored this way
  - » You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - » It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - Your data is already stored this way
  - You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - » Your data is already stored this way
  - » You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - » It will be slower in all ways

Pulled from here

Working with remote data

- We've only ever worked with local in-memory data
- We might want to work with remote on-disk data stored in databases
- Why?
  - » Your data is already stored this way
  - » You have so much data that it does not all fit into memory simultaneously and you need to use some external storage engine.
- If your data fits in memory there is no advantage to putting it in a database
  - » It will be slower in all ways

Pulled from here

dbplyr

- The most user-friendly companion for working with remote data
- You need to understand relational data
- It writes the SQL for you!
- . . . but it doesn't do everything
  - » mostly `SELECT` statements
  - » this will usually work for our purposes

dbplyr

- The most user-friendly companion for working with remote data
- You need to understand relational data
- It writes the SQL for you!
- . . . but it doesn't do everything
  - mostly `SELECT` statements
  - this will usually work for our purposes

dbplyr

- The most user-friendly companion for working with remote data
- You need to understand relational data
- It writes the SQL for you!
- . . . but it doesn't do everything
  - » mostly `SELECT` statements
  - » this will usually work for our purposes

dbplyr

- The most user-friendly companion for working with remote data
- You need to understand relational data
- It writes the SQL for you!
- . . . but it doesn't do everything
  - mostly `SELECT` statements
  - this will usually work for our purposes

dbplyr

- The most user-friendly companion for working with remote data
- You need to understand relational data
- It writes the SQL for you!
- . . . but it doesn't do everything
  - mostly `SELECT` statements
  - this will usually work for our purposes

dbplyr

- The most user-friendly companion for working with remote data
- You need to understand relational data
- It writes the SQL for you!
- . . . but it doesn't do everything
  - mostly `SELECT` statements
  - this will usually work for our purposes

Lingo

- **database backend** underlying software that manages the storage and retrieval of data in a database
  - » We'll use `RSQLite::SQLite()`
- **SQL** another, old programming language for managing relational data
- **SQL queries** our requests for subsets and organizations of the data to bring into local memory