QSS20: Modern Statistical Computing

Session 04: Merging and basic regex

Goal for next few sessions

- Some course housekeeping
- Exact matching: types of joins
  - Inner joins
  - Outer joins
  - Left joins
  - Right joins
- Basic regex for two purposes:
  1. Clean join fields for exact matching/merges
  2. Clean join fields for fuzzy/probabilistic matching/merges
- Fuzzy/probabilistic matching and merges

Goal for next few sessions

- **Some course housekeeping**
- Exact matching: types of joins
  - Inner joins
  - Outer joins
  - Left joins
  - Right joins
- Basic regex for two purposes:
  1. Clean join fields for exact matching/merges
  2. Clean join fields for fuzzy/probabilistic matching/merges
- Fuzzy/probabilistic matching and merges

PSET 1 Submission

- **Group**: tonight 1159 PM EST (will give extra time during break for u guys to meet and coordinate submission)
- **Individual**: same deadline unless using some/all of four free late days. If using all four, due at Saturday 04.25 at 1159 PM EST
- I'll give an emoji reaction on issue when i've seen it so you know it worked; i'll also comment on issue if i'm having trouble running your code

Organization of activity-based practice code

https://github.com/rebeccajohnson88/qss20_slides_activities#readme

These are jupyter notebook-based activities to practice Python or other concepts:

- 00_latex_output_examples_solutions.ipynb
  - **Data:** DC crime reports in 2020
  - **Concepts covered:**
    - Writing a pandas dataframe or table to use in LaTeX
    - Row filtering
    - Saving figures
    - Iterating and saving figures with informative names
- 01_pandas_datacleaning_examples.ipynb
  - **Data:** sample of Chicago health/hygiene inspection results
  - **Concepts covered:**
    - Cleaning column names (eg subbing out spaces and changing to lowercase)
    - Checking datatypes within a pandas dataframe and recasting
    - Creating new true/false variables using `np.where`
    - Creating new categorical variables that involve recoding an existing categorical variable using `map` and a dictionary
- 02_more_pandas_datacleaning.ipynb
  - **Data:** DC crime reports in 2020
  - **Concepts covered:**
    - Aggregation using `groupby` and `agg`
    - Lambda functions within aggregation
    - Recoding variables using `np.where`
    - Recoding variables using `np.select`
    - Recoding variables using `map` and dictionary
    - Loop to find matches within a broader pool of data
    - Function to find matches within a broader pool of data

Updated course schedule

https://rebeccajohnson88.github.io/qss20/docs/course_schedule.html

Tuesday 04-20	Intro to merging		Problem set one
Thursday 04-22	Merging: probabilistic merge and more regex		
Tuesday 04-27	Merging (continued) and PSET 1 review	Regular expressions for pattern matching	Final project step 1
Thursday 04-29	SQL via Python		
Tuesday 05-04	Text as data part one		Final project step 2
Thursday 05-06	Text as data part two		
Tuesday 05-11	TBD		
Thursday 05-13	Python: spatial data using geopandas		Problem set two
Tuesday 05-18	Python: reading data from APIs and basic web scraping		
Thursday 05-20	High-performance computing		Final project step 3
Tuesday 05-25	TBD		
Thursday 05-27	Workflow: Beamer and Tikz graphics		
			Slides for final

Steps towards final project

1. Make sure you can access this private repo and DM me if you need re-sent invite: https://github.com/rebeccajohnson88/qss20_s21_proj
2. Join `#sip_finalproject` on Slack
3. Will post details on Canvas tomorrow for `Final project Step 1`, due Tuesday 04.27 alongside the DataCamp assignment, but broadly: (1) sign up for background reading, (2) copy over LaTeX/Overleaf template I'll share, and (3) write <1 page memo outlining data used in the background reading, key takeaways, and interesting and feasible follow-up questions

Mid-term evaluation of our course

- Will circulate anonymous feedback survey later this week covering course pace, clarity, and what's going more versus less well

Goal for next few sessions

- Some course housekeeping
- **Exact matching: types of joins**
  - Inner joins
  - Outer joins
  - Left joins
  - Right joins
- Basic regex for two purposes:
  1. Clean join fields for exact matching/merges
  2. Clean join fields for fuzzy/probabilistic matching/merges
- Fuzzy/probabilistic matching and merges

Working example: have dataset on Dartmouth students and want to merge in background information about their district

- **Main or "left" dataset**

Student	Year	District	NCES ID
Rebecca	2021	New Trier High School	1728200
Jennifer	2022	Hanover High	3302670
Jason	2022	Homeschool	NA
⋮			

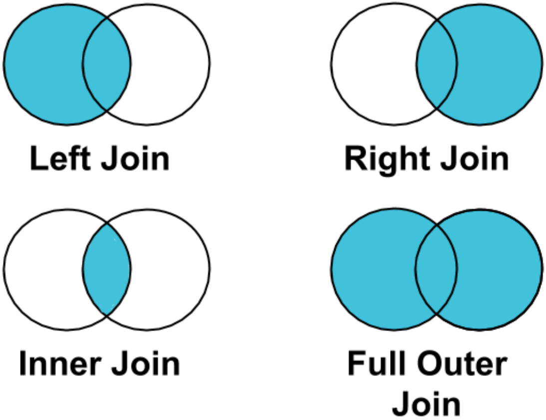
- **Auxiliary or "right" dataset**

District	NCES ID	% FRPL
New Trier HS	1728200	X%
Hanover HS	3302670	Y%
Lebanon HS	4107380	Z%
⋮		

Possible join keys

- **Unique identifier:** used for “exact matching”— or a Yes/No match on that basis
  - E.g., is the NCES ID of New Trier found in the dataset of demographics?
- **Other identifiers:** can be used for either “exact match” or for “probabilistic/fuzzy matching”
  - **Probabilistic:** what’s the likelihood that “New Trier district” and “New Trier HS” are the same entity?

Conceptual overview of four types of joins



Source: Trifacta

Inner join in this context

In words: "drop all students whose districts don't appear in the demographics data; drop all districts that don't appear in the Dartmouth student data"

- **Main or "left" dataset**

Student	Year	District	NCES ID
Rebecca	2021	New Trier High School	1728200
Jennifer	2022	Hanover High	3302670
Jason	2022	Homeschool	NA
⋮			

- **Auxiliary or "right" dataset**

District	NCES ID	% FRPL
New Trier HS	1728200	X%
Hanover HS	3302670	Y%
Lebanon HS	4107380	Z%
⋮		

Outer join in this context

In words: "keep all students from the student-level data; keep all schools from the school-level data; even if there's not an overlap"

Student	**Year**	**District**	**NCES ID**	**% FRPL**
Rebecca	2021	New Trier High School	1728200	X%
Jennifer	2022	Hanover High	3302670	Y%
Jason	2022	Homeschool	NA	NA
NA	NA	NA	4107380	Z%
⋮				

Left join in this context

In words: "keep all students from the student-level data; drop any school from the school-level data that doesn't merge onto a student"

- **Main or "left" dataset**

Student	Year	District	NCES ID
Rebecca	2021	New Trier High School	1728200
Jennifer	2022	Hanover High	3302670
Jason	2022	Homeschool	NA
⋮			

- **Auxiliary or "right" dataset**

District	NCES ID	% FRPL
New Trier HS	1728200	X%
Hanover HS	3302670	Y%
Lebanon HS	4107380	Z%
⋮		

Right join in this context

In words: "drop students who don't have a school in the school-level data; keep all schools from the student-level data even those that don't merge onto any student"

- **Main or "left" dataset**

Student	**Year**	**District**	**NCES ID**
Rebecca	2021	New Trier High School	1728200
Jennifer	2022	Hanover High	3302670
Jason	2022	Homeschool	NA
⋮			

- **Auxiliary or "right" dataset**

District	**NCES ID**	**% FRPL**
New Trier HS	1728200	X%
Hanover HS	3302670	Y%
Lebanon HS	4107380	Z%
⋮		

How do we code these different types of joins in practice? Example with left join and join key has same colname in both

```python

## perform a left join on the student data
## and schools data
stud_wschool = pd.merge(students,
                        schools,
                        how = "left",
                        on = "NCES ID",
                        indicator = "student_mergestatus")
```

- `how`: argument to tell it inner, left, right, outer, or cross; defaults to inner
- `on`: name of join key (in this case single key)
- `indicator`: optional arg to add a col to the resulting data (string is what to call it) that helps diagnose merge status; good for post-merge dx

Example with inner join and join key has different name

```
## perform a left join on the student data
## and schools data
stud_wschool = pd.merge(students,
                        schools,
                        how = "inner",
                        left_on = "NCES ID",
                        right_on = "ncesnumeric")
```

Example with left join and multiple join keys

```

## perform a left join on the student data
## and schools data
stud_wschool = pd.merge(students,
                        schools,
                        how = "left",
                        left_on = ["NCES ID",
                        "Dist name"],
                        right_on = ["ncesnumeric",
                        "distnamechar"],
                        indicator = "student_mergestatus")
```

Non-exhaustive checklist of merge diagnostics

1. How many rows were in each data before the merge? What about after?
2. If doing a left join, did we properly retain all left-hand side rows?
3. **For strings as join keys:** if a lot of rows were lost in a merge, could that be due to spelling/punctuation variations in a character join key?
4. **For numeric identifiers as join keys:** if a lot of rows were lost in a merge, could that be due to things like the id having leading zeros and those being stripped at some stage? (e.g., one dataset identifies an entity as 002548; another as 2548)

Next up: basic regex to improve match rates for strings as join keys

- In example below, what if we didn't have the NCES ID numeric identifier? Ways to improve match rates for spelling variations (sometimes called `entity resolution`)

Student	**Year**	**District**
Rebecca	2021	New Trier High School
Jennifer	2022	Hanover High
Jason	2022	Homeschool
⋮		

District	**% FRPL**
New Trier HS	X%
Hanover HS	Y%
Lebanon HS	Z%
⋮	