

QSS20: Modern Statistical Computing

Unit 05: Merging (exact)

Goals for today

- ▶ **Recap**
- ▶ User-defined functions: `lecture` & `activity` (from last week)
- ▶ Exact merging lecture
- ▶ Exact merging activity

Recap of LaTeX

What do you remember?

Recap of LaTeX

- ▶ Tips:

- ▶ Compile frequently
- ▶ Be sure to exit any environments you made (e.g., `itemize`)
- ▶ Mind the special characters:

escape character: backslash (`\`)

comment character: percent sign (`%`)

column separator: ampersand (`&`)

- ▶ Useful commands:

```
\begin{itemize} % start list
```

```
\begin{enumerate} % start NUMBERED list
```

```
\section{Title of section} \label{sec:shorthand}
```

```
\ref{sec:shorthand} % reference section in text
```

```
\begin{figure}
```

```
\includegraphics[scale = 0.5]{example_graphic.png}
```

What's missing from these code snippets?

Recap of shell & git

What do you remember?

Recap of terminal & git

► Tips:

- Get used to terminal AND git! Baseline for most real-world coding
- If "no file exists", double-check where you are and where the file is
- Install things from command line, NOT inside a notebook

► Useful commands:

```
pwd | cd | ls | cp | mv | rm # basic shell commands  
touch | mkdir | rm -r folder | rm *.png # also useful  
../ | ./ # up one level | current level
```

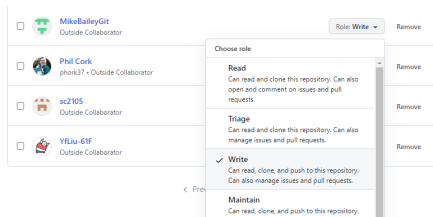
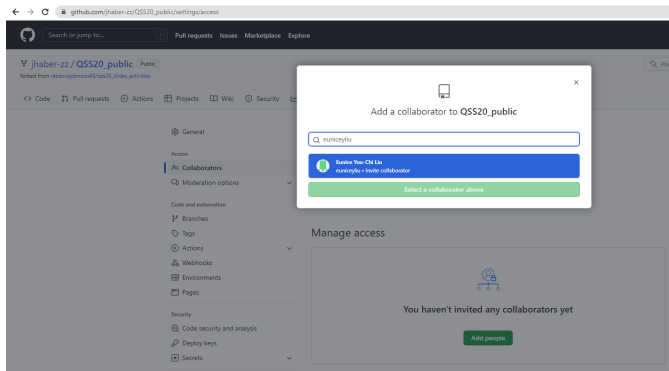
```
git clone | add | commit | push # basic git commands  
git rm | pull | status # also useful
```

Why use a '#' here, not a '%' as above?

How to collaborate on code with classmates

- ▶ Jupyter notebooks not an ideal collaborative tool, not built for version tracking
 - ▶ Don't allow simultaneous live editing, like google docs
 - ▶ Even with a shared virtual machine (e.g., Colab), interface is clunky: someone edits → everyone else immediately gets popup asking to overwrite their own version
- ▶ Suggestion 1: Live code collaboration sessions (at least two per pset)
 - ▶ Work through problems together in-person (ideally) or over zoom
 - ▶ Coordinate your schedules and plan ahead
- ▶ Suggestion 2: Work from **one group partner's** private git repository
 - ▶ Give the other person access and write permissions (see next slide)
 - ▶ Pass the editing baton back and forth, e.g.: "Hey, I'm done with section 1.3, want to pull my changes and start on 1.4?"
 - ▶ Once submitted, copy final version to other group repos (for reference)

How to give write access to your repo



Goals for today

- ▶ Recap
- ▶ **User-defined functions: lecture & activity** (from last week)
- ▶ Exact merging lecture
- ▶ Exact merging activity

Goals for today

- ▶ Recap
- ▶ User-defined functions: `lecture` & `activity` (from last week)
- ▶ **Exact merging lecture**
- ▶ Exact merging activity

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
⋮			

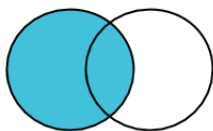
► **Auxiliary or “right” dataset**

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

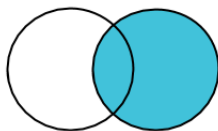
Possible join keys

- ▶ **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?

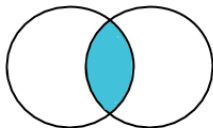
Conceptual overview of four types of joins



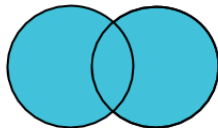
Left Join



Right Join



Inner Join



**Full Outer
Join**

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%
⋮		

DataCamp versus slide syntax

- ▶ DataCamp modules generally used this syntax for merges:

```
1 merged_df = df1.merge(df2 ,  
2                       how = "left" ,  
3                       on = "something" )
```

- ▶ Slides/solution code will tend to use this syntax:

```
1 merged_df = pd.merge(df1 , df2 ,  
2                       how = "left" ,  
3                       on = "something" )
```

- ▶ They produce identical answers so use whichever comes more naturally (I use latter because it's more similar to base R syntax)
- ▶ In addition, feel free to use self joins if useful but we won't be focusing a lot on those

How do we code these different types of joins in practice?

Example with left join and same colname for join key

1

```
2 ## perform a left join on the student data  
3 ## and schools data  
4 stud_wschoo = pd.merge(students ,  
5                          schools ,  
6                          how = "left" ,  
7                          on = "NCES ID" ,  
8                          indicator = "student_mergesource" )
```

- ▶ **how**: argument to tell it inner, left, right, or outer; defaults to inner
- ▶ **on**: name of join key (in this case single key)
- ▶ **indicator**: optional arg to add a col to result (default name "_merge") indicating source ("left_only", "right_only", or "both"); helpful for merge status & post-merge diagnostics

Example with inner join and different colname for join key

```
1  
2 ## perform a left join on the student data  
3 ## and schools data  
4 stud_wschoool = pd.merge(students ,  
5                         schools ,  
6                         how = "inner" ,  
7                         left_on = "NCES ID" ,  
8                         right_on = "ncesnumeric")
```

Example with left join and multiple join keys

```
1  
2 ### perform a left join on the student data  
3 ### and schools data  
4 stud_wschoool = pd.merge(students ,  
5                           schools ,  
6                           how = "left" ,  
7                           left_on = ["NCES ID" ,  
8                                       "Dist name" ] ,  
9                           right_on = ["ncesnumeric" ,  
10                                      "distnamechar" ] ,  
11                          indicator = "student_mergesource" )
```

Example with left join, multiple join keys, and some overlapping, non-join columns that we want to differentiate

```
1  
2 ## perform a left join on the student data  
3 ## and schools data  
4 stud_wschoool = pd.merge(students ,  
5                           schools ,  
6                           how = "left" ,  
7                           left_on = ["NCES ID" ,  
8                                       "Dist name"] ,  
9                           right_on = ["ncesnumeric" ,  
10                                      "distnamechar"] ,  
11                           indicator = "student_mergesource" ,  
12                           suffixes = ["_students" ,  
13                                       "_schools"] )
```

Non-exhaustive checklist of merge diagnostics

1. How many rows were in each dataset 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 or lagging 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%
⋮	

Where we are

- ▶ Recap
- ▶ User-defined functions: `lecture` & `activity` (from last week)
- ▶ Exact merging lecture
- ▶ **Exact merging activity**

- ▶ `public_data/sd_df.csv`: sample of business tax certificates for San Diego-based businesses—each row represents one unique business; cols for industry (6-digit NAICS code)

account_key	dba_name	council_district	naics_code	naics_description	naics_
1974000448	ERNST & YOUNG LLP	cd_1	541211	OFFICES OF CERTIFIED PUBLIC ACCOUNTANTS	
1974011093	HECHT SOLBERG ROBINSON GOLDBERG & BAGLEY LLP	cd_3	5411	LEGAL SERVICES	
1978039819	RSM US LLP	cd_1	541211	OFFICES OF CERTIFIED PUBLIC ACCOUNTANTS	
1978042092	THORSNES BARTOLOTTA MCGUIRE LLP	cd_3	5411	LEGAL SERVICES	
1979046817	KORENIC & WOJDOWSKI LLP	cd_7	5412	ACCOUNTING/TAX PREP/BOOKKEEP/PAYROLL SERVICES	

- ▶ `public_data/naics_df.csv`: exhaustive listing of all 6-digit NAICS codes from the Census Bureau with added information

naics	naics_description
111140	Wheat Farming
111160	Rice Farming
111150	Corn Farming
111110	Soybean Farming
111120	Oilseed (except Soybean) Farming

- ▶ **General goal:** match the two to investigate things like which industries are *not* represented in San Diego small businesses

Outline of notebooks

- ▶ Blank notebook to work on with partners/small group: [04_merging_exact_blank.ipynb](#)
- ▶ Input data: `public_data/sd_df.csv` and `public_data/naics_df.csv`
- ▶ Solutions notebook to review after class (please don't look during activity!): [04_merging_exact_solutions.ipynb](#)
- ▶ If interested, notebook where data was cleaned/prepped to make easier to analyze: [04_helper_merging_dataprep.ipynb](#)