PPOL 564: Data Science I

Bonus session: Probabilistic record linkage

How does this fit in with content we covered?

- 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: which businesses received PPP loans?

Focal dataset: sample of PPP loans for Winnetka businesses

Business name	NAICScode	City	State	Zip
CLASSIC KIDS, LLC	541921	Winnetka	IL	60093
NORTH SHORE COUNTRY DAY SCHOOL	611110	Winnetka	IL	60093

Other data:

Business name	City	State	Zip
CLASSIC KIDS	Newport Beach	CA	92660
CLASSIC KIDS UPPER WEST	Manhattan	NY	10024
CLASSIC KIDS	Winnetka	IL	60093
CLASSIC KIDS PHOTOGRAPHY	Chicago	IL	60614

What's the role of fuzzy/probabilistic matching?

- **Exact match:** would find no matches in previous example since there's no Classic Kids, LLC in the Yelp data; `pd.merge` fails us
- **Probabilistic match:**
  1. Compares a given pair of records
  2. Using 1+ fields—e.g., business name; zip code; address—what's the probability that the pair is a match?

General workflow for probabilistic matching, regardless of package

1. **Preprocess the relevant fields in the data:** none of these algorithms are magic bullets; each can have significant gains from basic string preprocessing of the relevant fields (e.g., should we remove `LLC`?; how are street addresses formulated)
2. **Decide if/what to "block" or exact match on:** when creating the candidate pairs, what's a *must have* field where if they don't match exactly, you rule out as a candidate pair?
   - **How do you decide this:** fields that are more reliably formatted (e.g., two-digit state)
   - **Main advantages:** potentially reduces false positives; reduces runtime/computational load
3. **If blocking, creating candidate pairs based on blocking variables:** if we blocked on state, for instance, this would leave the two IL businesses as candidate pairs for our focal business
4. **Decide on what fields to match "fuzzily":** these are things like name, address, etc. that might have typos/different spellings. The two components are:
   - How to define similarity: string distance functions
   - What threshold counts as similar enough
5. **Within candidate pairs, look at those fuzzy fields**
6. **Aggregate across fields to decide on "likely match" or "likely not"**

Specific workflow depends on (1) manual versus (2) package

1. In activity code, we'll (1) first do things manually and then (2) use a package
2. Packages in Python:
   - `recordlinkage`: focus of example code
   - **Others:** `fuzzy-matcher`; `sklearn` if we have a small set of "true matches" and want to build a model that predicts matches
3. Packages in R: `fast-link`; `RecordLinkage`

Guide to data and notebooks

1. **Workbook with example code:**
   https://github.com/rebeccajohnson88/PPOL564_slides_activities/blob/main/activities/fall_22/solutions/13_probrecordlink_codesample.ipynb
2. **For your practice:**
   - Blank notebook: https://github.com/rebeccajohnson88/PPOL564_slides_activities/blob/main/activities/fall_22/13_probrecordlink_blank.ipynb
   - Solutions: https://github.com/rebeccajohnson88/PPOL564_slides_activities/blob/main/activities/fall_22/solutions/13_probrecordlink_solutions.ipynb
3. **Datasets:**
   - `sd_forfuzzy.csv`: sample of businesses from San Diego tax certificate data used in exact merging activity
   - `pploans_forfuzzy.csv`: sample of businesses receiving federal PPP loans