Modern Record Linkage: STA 794

Assignment 2: Fall 2016

Due Friday November 4 at 5:00 PM
Late assignments will not be graded. Submit early.

Show all your work and calculations to receive full credit on each problem. Simply stating the solution without work will be worth zero points. Please make sure all of your work is fully reproducible either in R Markdown or in your programming language of choice.

The goal of this first assignment is getting familiar with record linkage tasks using the Record Linkage package in `R`.
Create four tasks or directories (a task is defined below) called

1. `generate-raw/`
2. `blocking/`
3. `feature-generation/`
4. `classify/`

Each task should have a structure with an `input/`, `output/`, `src/` directories. Note that for some of these tasks, these directories may be empty. Let's walk through what each of these tasks should look like.

Suppose that we are working on a task called `my-new-task/`.

1. `input/`
   The `input/` directory contains any files that src/ will call from any functions that you write for the task `my-new-task/`.
2. `src/`
   The `src/` directory contains any code that you write for the task `my-new-task/`. All output should be written into `output/`.
3. `output/`
   The `output/` directory contains any code that is output for the task `my-new-task/`.

As an example: the task `generate-raw/` should generate data. For this assignment, please generate output data from the RLdata500 package. To help you with this, I have written the first task for you, which you can access from the public repo: `https://bitbucket.org/resteorts/reclink/` by cloning the repo. (For more information on how to clone a repo, please see: `http://www2.stat.duke.edu/~rcs46/git.html`).

Specifically, to clone the repo, just use "git clone https://bitbucket.org/resteorts/reclink/"
Let's now talk about what each task should do.

1. `generate-raw/`: This task generates data (which has already been done for you). The output data lives in `output/`

2. `blocking/` This task will take as an `input/` the data from `generate-raw/`. The `src/` file should perform multiple types of blocking.

   - all-to-all record comparisons
   - considering a deterministic feature as a block and then performing all-to-all record comparisons
   - using the features of the records and the soundex as blocking criterion

   For each blocking rule, return three files called

   (a) `pairs-all-comparisons.csv`,

   (b) `pairs-determined-comparisons.csv`,

   (c) `pairs-soundex-comparisons.csv`.

   In each file, return

   (a) all pairs of records that fall into a block.

   (b) In addition, make sure to include a column of match/non-match, where 1 refers to a match and 0 refers to a non-match.

   (c) Make sure that each file has a header and you separate each element in the file with a `|` delimiter.

3. `feature-generation/` This task will take as an `input/` the output data from `blocking/`. The `src/` file should

   (a) Take each pair of block indices

   (b) Find the corresponding features

   (c) And find a similarity comparison for these features under the jarowinkler metric.

   This should be done for all pairs in the blocking files. There will be three sets of output files, namely

   (a) `feature-comparison-all-comparisons.csv`,

(b) `feature-comparison-determined-comparisons.csv`,

(c) `feature-comparison-soundex-comparisons.csv`.

Think carefully about what this file should contain before writing it to output (and also look at the classify step).

4. `classify/` Given the three output files from the task `feature-generation/`, apply the following the Fellegi Sunter method such that it returns

   (a) index1

   (b) index2

   (c) probability of classification

   (d) match/non-match

   Also, code up at least one classification or clustering method (Bayesian) or non-Bayesian of your choice. You may use packages or existing code if they exist. Examples include:

   (a) random forests

   (b) support vector machines

   (c) the method of Goldstein et al (2016)

   Place your final output in `output/classify-fs.csv` and `output/classify-random-forests.csv`

   Finally, compute the recall and precision in `output/classify-fs.csv` and `output/classify-random-forests.csv` and report your findings.

   Note: for help in coding up the Fellegi-Sunter method, I recommend looking at Murray (2015) and Murray's code, which may help you. `http://repository.cmu.edu/jpc/vol7/iss1/2/`.