Matching Explanation Connor Galvin

Algorithm Explanation

To match my data, I originally planned to use a simple algorithm. I was going to look at each book name and stop when I hit a parenthesis. I thought this was the only difference between book names. Some books are part of series, and the website (good reads) adds a series grouper into the title surrounded by parentheses. However, after closer inspection I found that there were other differences in book names. For example, good reads uses the international name for Harry Potter and the philosophers stone while the other site uses the sorcerer's stone. After acknowledging this I searched for python packages that could help with fuzzy matching. I found a package called fuzzy matcher, that uses SQL light's string-matching algorithm in combination with probabilistic record linkage. Fuzzy matcher takes two data frames and the comparison columns. I found that only comparing book names produced a better result due to duplicate author names. After running fuzzy matching, I filtered the data frame to remove all results where the author's name did not match.

Problems

I found that fuzzy matching's probability score decreased as the book titles length went down. This led to correct short title matches being mixed in with wildly incorrectly matched longer titles. I also found that while one website pieced out each series book names the other would group them all together. This led to less precise matching for series books. The python package fuzzy matching has also been discontinued. This leads me to believe that in the future I should use a more up to date matching package and compare its results to the older python package fuzzzymatching.

Sizes

Table A 2500 books

Table B 2400 books

Cartesian Product = 6000000 total tuple pairings

Table C 508 total number of tuples

Cleaning

To clean my data, I had to first find a replace all author names that had an additional title of (goodreads author) added into the name. Thankfully this was easily done in excel.