csae CENTRE FOR THE STUDY OF AFRICAN ECONOMIES

UNIVERSITY OF OXFORD

HOW TO MERGE DATASETS USING THE MATCHIT ROUTINE

Suppose you set out to conduct your study; this time a randomised experiment with two waves of data collection, baseline and endline. However, you face a common data challenge where the unique identifiers in the baseline are erroneously missing or wrongly captured such that no successful merging is possible. Luckily, you had the names of your subjects captured in two variables as "`surname`" and "`firstname`".

Here is a go-around to get your data merged using the "`Matchit`" routine[1].

1. Datasets

There are two datasets to merge: `baseline` and `endline`.

Each dataset has the subject names under "`surname`" and "`firstname`" variables. The names are also irregularly entered. That is, in the baseline data, a given surname is different from the one entered in the endline. The 'supposed' unique identifier "pid" does not uniquely identify the observations either.

2. The Matchit routine

a. You will need first to concatenate the two names to create on variable such that, if the surname is John and the first name is Doe, the new variable has `JohnDoe`.

```
*baseline
use baseline, clear
egen namesite = concat(firstname surname)

*endline
use endline, clear
egen namesite = concat(firstname surname)
```

b. With irregular `pids` in both datasets, you are set to attempt a fuzzy merge. This routine tries to match text and then generate a similarity score such that $0 \leq similscore \leq 1$. A similarity score of 1 implies a perfect match between texts found in both the datasets. We can proceed as follows.

```
*Matchit routine
use baseline, clear
matchit pid namesite using endline.dta , idu(pid) txtu(namesite)
```

[1] Julio Raffo, 2015. "MATCHIT: Stata module to match two datasets based on similar text patterns," Statistical Software Components s457992, Boston College Department of Economics, revised 20 May 2020.

c. You can now get a sub-dataset of perfect matches `(similscore == 1)` and save that as a separate dataset. Note that all this code has done is create a similarity score variable based on the imperfect `pid` and `namesite` variables, then patched the two variables into your master dataset (baseline).

   If you just wanted to get a final dataset of perfect matches only `(i.e. similscore == 1)`, all you need to do is to drop all other observations with `similscore` < 1.

```
*Now keep Exact score (similscore == 1) as separate dataset


gsort -similscore // here we arrange the similscores in descending order

drop if pid == pid[_n] & similscore != 1 // note that we want to remain with
observations that uniquely matched and with a similarity score of 1.
save exactmatch_baseline, replace
```

d. Next, we merge the bridge set of exact matches back to the baseline file.

```
merge 1:1 pid namesite using baseline
drop _merge
save baseline_1, replace
```

e. We have to repeat the same process for the endline data.

```
*Repeating with the endline data

use endline, clear
matchit pid namesite using baseline.dta , idu(pid) txtu(namesite)
gsort -similscore
drop if pid == pid[_n] & similscore != 1
save exactmatch_endline, replace

merge 1:1 pid namesite using endline
drop _merge
save endline_1, replace
```

f. The last step is to now merge the two sets of perfect matches (baseline_1 and endline_1)

```
use baseline_1, clear
keep if similscore == 1
gsort namesite
save baseline_final, replace

use endline_1, clear
keep if similscore == 1
gsort namesite1
save endline_final.dta, replace
merge 1:1 namesite using baseline_final
drop _merge

save final_data, replace
```

The most interesting part is that the matchit algorithm gives you scores of matches that are less than 1. Indeed, a normal merge could give you the `final_data` achieved above. Instead, one can first eliminate the observations that match 100% in both datasets thereby remaining with datasets that do not match perfectly. Here, you can choose what levels of match are more sensible to try given that each observation in the master dataset is essentially iterated through every observation in the using dataset to generate a match. That means, one observation can have several matches even though the largest score is essentially more plausible as a good match. To fix the match at some probability score, assume you want matches that are greater than or equal to 0.7 in similarity scores. The code to attain this is;

```
matchit pid namesite using endline_70.dta , idu(pid) txtu(namesite) t(.7)
```

The final step will be a simple append routine to get a composite project data with the bits of data tied back together. A downside is that you will end up generating quite a number of sub data files depending on the match levels you iterate.

Note that `Matchit` is a routine that can be deployed for a number of data consolidation and cleaning tasks. It is worth checking some more of what `Matchit` can do here:
https://www.stata.com/meeting/switzerland16/slides/raffo-switzerland16.pdf.

Some of the highlights are that with cleaner datasets, you can use Matchit's powerful inbuilt weights option to penalize some of the text that is more frequent by giving lower scores. Text that is less frequent is then given higher scores meaning that the routine is guided to produce better similarity accuracies. Another useful option that accords faster computational speed (particularly in large datasets) removes redundant information and reduces the size and depth of index. This option works optimally if you have spent some time to clean the variables of interest as much as you can.

George Kinyanjui, Doctoral Fellow, University of Cape Town
geejoekaris@gmail.com | knygeo002@myuct.ac.za
04 February 2021