# CODERS' CORNER

CENTRE FOR THE STUDY OF AFRICAN ECONOMIES

## LARGE DATA SETS AND THE NEED FOR SPEED

Very large datasets slowing down even the simplest computations in Stata? Sound familiar? I recently ran into an issue where executing several loops to generate a large number of dummy variables (so not a very complex task) would take 24 hours. The following blog post discusses a few tricks on how to improve the speed of Stata jobs when handling large data sets.

The reason why Stata becomes slow when handling large data sets is that it loads the full data set into memory when working with it. Obviously, a lot of Stata users have experienced similar problems before, so there are a fair number of ideas aimed at improving computational speed floating around online.

Little things that can improve computational speed and/or converse memory are…

- … the Stata version used: Stata MP can be an advantage when working with large data sets as it exploits available capacities across multiple cores for computationally intensive tasks
- … commands used: Some commands are faster than others, e.g. reghdfe is considerably faster than xtreg (see related coders' corder post)
- … variables: some variables might take up more memory than necessary, e.g. string variables that accommodate a lot more characters than actually required.

**compress** is an amazing command, which converts variables to the smallest storage type possible without losing any information. I'd recommend executing it each time before saving a data set or even before running a major command like reshape or a long loop.

The trick that ultimately cut the time to run my do-file from **24 hours to 3 hours**, however, was to slice the data set into small data sets (in my case 40) and thus considerably reduce the amount of data in memory. With less data in memory the time required to execute each of the required commands was reduced substantially, e.g. from half a second to generate an indicator variable to a fraction of a second.

**In short:** Cut your data into several smaller data sets, run your code, append the datasets back together.

### Here is an example for the code I ran:

use *fulldataset.dta* \\\ *load full data set*

levelsof *identifier*, local(*clustername*) \\\ identify a suitable way to split your data set and identify all possible values it can take (e.g. split by province, by cohort)

foreach *cluster* of local *clustername* { \\\ start a loop over all possible values (e.g. provinces)

use *fulldataset.dta* \\\ re-load dataset with all observations

keep if *identifier* ==" ` *cluster*' " \\\ only keep observations for one province, cohort etc.

**run your code** (e.g. gen warm= temperature>25)

tempfile `*nametemporaryfile*' \\\ save data in temporary file

save ` *nametemporaryfile*', replace

if *identifier* == "firstcluster" { \\\ start a new file with observations from the first cluster

save "*newfile*", replace

}

else{ \\append other observations to it

use "*newfile*"

append using `*nametemporaryfile*', force

save "*newfile*", replace

}

}

The idea is borrowed from a Stata command called *parallel*, which implements many of the above described tricks in a user-friendly way. Unfortunately, in my experience, the command struggles with reliably appending the split data set back together after implementing the core job for very large data sets, but this is definitely also a command that you could consider for this kind of task

**Reference:**

George Vega, 2013. "Introducing PARALLEL: Stata Module for Parallel Computing," 2013 Stata Conference 4, Stata Users Group.

**Verena Wiedemann, DPhil Candidate in Economics, St Antony's College, Oxford**
**14 October 2019**