Applied Tips for Applied Micro RAs*

Last updated: September 6, 2022

Communication

1. Reply all. Don't forget to always "reply all" in your emailing!
2. Take notes (always, and on everything). Remember to always take notes during meetings. Make sure to note down everything that could be useful later.
3. Keep an updated to do list. Keep an updated to do list that you are ready to share. Your team will want to know what you are doing, what you have done since you last spoke with them, and what you are planning to get done by when.
4. Provide context. Always provide all the necessary information to understand what you are talking about. Most likely, no one else in the team will have thought about the issues you are bringing forward as much (or as recently) as you have.
5. Be specific. E.g. "Should I drop partially incomplete interviews?". It may be hard to answer this question without more information on:
   (a) how many observations you would drop and over how many;
   (b) what "incomplete" means in case at hand;
   (c) why they might be incomplete.
6. If in doubt, double check your task. If you are not sure how to interpret the instructions given, ask again especially before embarking into a time consuming task.
7. Reach out if you think something is infeasible. You may be given tasks that are not feasible. Your PIs are far from omniscient (exceptions do not apply!) and may underestimate the time or resources needed for a task. If you think this is the case, let them know. Tasks may be infeasible because of time, or for other reasons (e.g., the variable they ask for does not exist).
8. Present your work in a clear way. Format your tables in a clear and legible way, and be as clear as possible in your exposition.
9. Add notes below tables and figures which specify the unit of observation, the sample, and the model specification you used.

*Authors: Livia Alfonsi (Department of Agricultural and Resource Economics, UC Berkeley) and Gaia Dossi (Department of Economics, London School of Economics and Political Science). We thank Davide M. Coluccia, Amanda Dahlstrand-Rudin, Luca Favero, Nicola Fontana, Gaia Gaudenzi, Louise Guillouët, Sean Higgins, Peter Lambert, Emilia Tjernström, and Arthur Turrell for their feedback and suggestions. Please use this form if you wish to submit feedback or new tips.

10. Add a summary of key takeaways. When preparing a note, add a list with the most important points at the beginning of the document.

11. Ask for feedback, especially early on. This is helpful to ensure you are organizing and communicating the work effectively.

Organization

1. Work on Dropbox (or any other file sharing platform used by your team). Do not work offline and then upload your work in order to avoid creating conflicting copies.

2. If working with sensitive data, use encrypted folders. Use a strong password, which will be passed on between generations of RAs and known by professors. Avoid Dropbox in this case.

3. Make sure to be in sync! Always double check you are working on the most updated version of a dofile or document.

4. Close files when you are done. This minimizes the risk of creating a conflicting copy.

5. Keep track of your data sources. Every step you make needs to be replicable, so if you bring in something new to the process (e.g., a new dataset from the web or from another project) make sure to explain where it came from exactly. *Exactly* means that anyone should be able, by following your steps, to recreate exactly the same dataset. You want to set up clear and updated README files for this.

6. Organize dofiles by type. Keep separate dofiles: (i) for cleaning the raw data, (ii) for merging and getting the datasets ready for the analysis, (iii) to create tables and figures.

Working with Data (Stata tips)

Code

1. Always use / instead of \. This makes sure that your code runs on any operating system.

2. Break long lines of code. This makes your code more legible. Use /// to break your lines.

3. Provide key information at the beginning of your dofile. You should report the author of the dofile, the date when the dofile was created or substantially modified, and what the dofile does in a nutshell, as well as the name of all the INPUT and OUTPUT .dta files.

4. Use directories and set paths through macros. Set up your own directory only once, at the beginning of the dofile. In this way, everyone in the team will be able to run every dofile easily. Also check out the *dropbox* command!

5. Comment your code. Add comments before each couple of lines of code in order to explain in plain words what you are doing. In this case *more* is more!

6. Run each dofile from beginning to end. A dofile needs to run from the beginning to the end, always. Avoid running only parts of it.

7. Do not overwrite existing datasets. Each time a new dataset is generated, this should have its own name (i.e. avoid generating the same dataset in two different parts of your code).

8. Save your code often. If Stata crushes, you will not be able to retrieve unsaved work. Also remember that if you close the main Stata window, the dofile window will close as well.

9. Install new commands using *ssc install*. User-generated commands are not automatically available. To use the command *cmd*, run a one-off installation by running the line *ssc install cmd* in the command window. Make sure to keep track of all the user-generated commands used in the code, they will come in handy for the replication package.

10. Google, Google, Google. If you are stuck, try googling your doubt. There are high chances that someone else will have run into the same doubt, and that there is documentation online.

11. If you are still stuck... after several hours of googling, check out this very useful STATA command: *motivate*.

Data

1. Start from the very raw data. Import any new dataset as you downloaded it or as you received it, without making any manual edits.

2. Never erase or modify the raw data. Never use the Stata *edit* command to change the data.

3. Check for outliers. When working with a variable for the first time, check if there are outliers. The most straightforward way is to use the command *summ varname, d*. You can also plot the density of *varname* with the command *kdensity varname*.

4. Check for duplicates. Look up the commands *unique*, *duplicates*.

5. Clean missing values. When using a variable for the first time, check its codebook and recode all missing values (99, 999, -1, or similar) replacing them as missing (i.e., ".").

6. Take care of missing values when you generate a new variable. Make sure you understand how missing values are treated by the commands you use. A few examples below:

   - When creating a dummy, *gen* and *gen byte* treat missing in different ways:

     (a) *gen female=1 if gender==1*
     replace female =0 if gender==0

     (b) gen byte female=(gender==1)

     If *gender* is missing, *female* will be missing in case (a), it will be "0" in case (b). The same applies when you define new variables using inequalities.
   - *collapse (sum)* treats missing as zeros.
   - Many commands have specific *missing* options (e.g. *rowsum* and *rowsum* with option *missing* will give you different results).

7. Look at the data. Every time you create a new variable, look at it. You can look at the raw data using the command *browse*. Alternatively, you can produce a frequency table for the variable *varname* using the command *tab varname, m*. This is to double check that the result is consistent with what you expected (e.g. if you are creating a dummy and from the tab you have a variable that takes only values "0", something might have gone wrong).

8. Justify why observations do not merge. When you use the *merge* command make sure to know what happened with the observations that are not merging. And always copy the output of the merge on the dofile.

9. Never use *merge m:m*. If you really feel that's the only thing you can use, try to think through other alternatives (*merge 1:1, merge m:1, merge 1:m, joinby*) and why they are not suitable.

10. If you can, do it with a loop. If you are writing repetitive code, check whether it would be possible to make it more concise by using a loop.

Large Datasets

1. Avoid strings. Storing strings requires more memory than storing numerical variables. If possible, use value labels instead.

2. Drop all variables you do not need. This will save you a lot of memory (and computing time)!

3. *reghdfe* is useful to run regressions with large datasets. This command allows you to absorb more than one set of fixed effects, and to cluster standard errors by more than one dimension.

4. *chunky* speeds up the import of very large text data.

Additional Resources (Organization)

- A Guide to RAing in Economics
- Pre-doctoral Memos
- Code and Data for the Social Sciences

Additional Resources (Programming)

- A Guide for Coding for Economists
- Python for Research Projects
- R for Research Projects
- Programming in Stata
- Twitter Thread on Useful Stata Tips