Practitioner's Corner

Christopher R. Knittel and Konstantinos Metaxoglou[a,*]

Working with Data: Two Empiricists' Experience

DOI 10.1515/jem-2016-0001

Abstract: We propose a set of best practices on how to organize empirical research drawn from our experience. We offer some ideas on organizing, processing and analyzing data efficiently with an eye towards quality control, documentation, and replicability. Although these best practices are by no means unique, they have served us and colleagues well over the years. We hope they will be helpful to students and young economists in their research endeavors.

Keywords: best practices; data; documentation; empirical research.

JEL Codes: A22; A23; A29.

1 Introduction

Producing high-quality empirical research is a difficult task and requires a lot of skill and talent. The objective of this paper is to provide some guidance to young economists working not only in academia, but also in research institutes, government agencies, and the private sector, on how to streamline many of the tasks involved in empirical research. Ranging from general organizing principles to specific coding advice and tricks, the practices we propose here are by no means the only way to organize empirical research. They happen to have worked well for us and we hope they will work, in whole or in part, for the reader as well.[1]

There are many and good reasons to stay organized while conducting empirical research regardless of whether it is academic research or not. In academia, very few papers are accepted after the first submission. Most papers go through a revision process that may span several months or even years. In the meantime, you have devoted a substantial amount of time to another project(s) and all the familiarity you had with the details of the paper that is now a "Revise & Resubmit" has worn off. Being organized can be very helpful to respond to referee requests regarding modifications of the existing or additional analysis. In addition, many journals, such as those published by the American Economic Association, require authors to disclose their code and data for the purpose of replicability; replicability on its own is of paramount importance in academic research. Another reality in the economics profession is that there is a large number of papers that are products of collaboration. Collaboration is also a very good reason to stay organized, especially when the collaborators work together for the first time. A few ground rules, which may be as simple as how to store and

1 See also the Practitioner's Guide by by Gentzkow and Shapiro (2014), which also touches on many themes discussed here. Ball and Medeiros (2012) offer a protocol for documenting data management and analysis.

[a]Prior to joining the Faculty at Carleton, Metaxoglou spent 7 years with an economic consulting firm working on Antitrust litigation. Some of the practices described here come from his consulting experience. We have not received any financial benefit for promoting the software discussed here. The usual disclaimer applies.

***Corresponding author: Konstantinos Metaxoglou,** Carleton University – Economics, 1125 Colonel By Drive, Ottawa, Ontario K1S 5B6, Canada, E-mail: konstantinos.metaxoglou@carleton.ca

Christopher R. Knittel: William Barton Rogers Professor of Energy Economics, Sloan School of Management, Center for Energy and Environmental Policy Research, MIT and NBER, Cambridge, MA, USA

process the raw data, organize the analysis, and maintain version control of the manuscripts, make the collaboration smooth and very efficient. Conducting empirical research outside academia entails a similar final product, such as a report by a central bank or a government agency, which reaches its final shape following a process very much like that of an academic paper. In the private sector, the fast pace, client budgets, and tight timelines are all additional reasons to be organized in preparing an expert report or a white paper. In the litigation context, an opposing economic expert and his team now play the role of the referee. The workload and time constraints of the private sector make collaboration a rule and not an exception. In short, the best practices we describe in this paper can be useful to all empirical economists.

In the remainder of the paper, we describe these practices starting with some recommendations about software in Section 2. We will then discuss organization of electronic workspace, largely referring to folder structures in the hard drive you own or in the hard drive that you rent from somebody else in the cloud, in Section 3. This will serve as a skeleton for the themes discussed in Sections 4–9, which span topics ranging from handling raw data to writing the first draft.

2 Software

We do not attempt to convince users about which operating system (O/S) they should use. Some use Windows and others use Mac OS but they can still work together without (too many) arguments; at least, this is the case for us. Additionally, the software discussed here is available for both PCs and MACs.

Regardless of the O/S of your choice, Cloud computing has revolutionized the way we store, access, and share files. We use Dropbox to perform all of these tasks. It is a very convenient storage system in the cloud that allows multiple users to have access to the most updated material without needing to move and update files manually increasing the risk of mistakes caused by outdated files. It offers multiple subscription plans that are reasonably priced. In its desktop application, Dropbox shows up on the user's computer as any other folder. The files stored in it will be updated automatically as long as the computer is connected to the internet. There is also a cellphone App that allows the user to see the content of a Dropbox folder. Dropbox also eliminates the need for backups as this is an automated process if you subscribe to the appropriate plan. With the recent advances in cloud computing, there are many alternative products, such as Google Drive and OneDrive. Some of these products are available at no cost.

Once the research question and the testable hypotheses are framed, working on empirical projects usually entails the following steps: (i) obtain raw data, (ii) prepare the data for subsequent analysis, (iii) perform the analysis, (iv) write the paper that reports the findings. Raw data can be obtained through many sources including the Web. The wealth of data available on the Web is enormous and knowing how to access them efficiently is extremely useful. For simple HTTP requests we use the LWP functions in Perl. In a remarkably simple syntax, Perl allows the user to fetch and store a large number of file types posted on the Web, especially when the link addresses are organized in a consistent fashion.[2]

We use mostly, if not exclusively, Stata to process raw data and prepare them for analyses. Stata allows the user to import data from multiple formats (e.g. CSV, TXT, XLSX) using a few simple commands and a very friendly environment. SAS is also a useful tool for processing data and it can manage large datasets more efficiently than Stata but it is significantly more expensive and intimidating to the new user. Stata is also able to export data into multiple formats including Excel, which we find very useful to do basic data exploration.

Pivot tables and charts in Excel are particularly useful for (preliminary) data exploration and analysis. They allow the user to review and analyze data generating nice visuals and tabulations with a large degree

2 For example, using `getstore` "http://www.eia.gov/electricity/monthly/archive/pdf/02261001.pdf", " MER-2010-01.pdf", the user can fetch the PDF document associated with the January version of the US Energy Information Administration (EIA) Monthly Energy Review (MER) and store it as MER-2006-01.pdf. Similarly, using `getstore` "http://www.eia.gov/electricity/monthly/archive/pdf/02261002.pdf", "MER-2010-02.pdf", the user can fetch the PDF document associated with the February version of the EIA MER. Python also has similar functions. Both Perl and Python are free with a very large community of users and an endless amount of resources on the Web.

of flexibility. We do not advice using Excel for analyses that require formulas for a very simple reason: it is extremely difficult to audit an Excel spreadsheet. The pivot tables and charts, however, do not need to rely on any formulas as they can be based on data exported directly from some other software, such as Stata. You can use macros in Excel to automate processes such as creating tables with summary statistics and regression results or producing figures, which can then be imported into drafts.

Stata and MATLAB are two very powerful tools for econometric analyses. Stata is particularly friendly and well suited for "off-the-shelf" estimation because it has a vast number of built-in commands and estimation routines for cross-sectional, panel, and time-series data. In addition, it has a very large community of users that contribute their own commands and estimation routines that can be very easily accessed from its command line.[3] Stata has also its own journal and it is the software used in very popular undergraduate and graduate econometrics textbooks such as Wooldridge (2013) and Cameron and Trivedi (2005). Many of our colleagues would argue that SAS is also very powerful and so is Eviews when it comes to time-series econometrics. Others will also advocate for R, which is available for free. MATLAB can handle better more customized exercises, such as simulations in industrial organization models, nonlinear optimization, solution of general equilibrium models, to name a few examples. Similar to Stata, both R and MATLAB have a very large community of users contributing to their very rich collections of functions with centralized locations.[4] Gauss and Octave are two alternatives to MATLAB. Dynare is a very good complement for solving dynamic stochastic general equilibrium models in macroeconomics. For text mining, when needed, we use Perl and Python leveraging their enormous capabilities to access content in PDF files.[5] With the exception of Perl, Python, R, Octave, and Dynare, the software packages discussed here require some sort of licensing, which in turn may include annual maintenance fees. Our view is that not a single piece of software is equally good in everything.

We do most of our writing in LaTeX, which is the gold standard for professional typesetting of academic books and journals. It is extremely powerful, with endless resources at the Web, and it is free.[6] One of its big advantages, which will become more clear below, is its seamless interaction with PDF files. And since nothing is perfect, one of its big disadvantages is that it is extremely painful and time consuming to create tables. However, we have a trick around this in a subsequent section. The vast majority of users exposed to it hate it the first week and keep wondering why they should ever give up on Microsoft Word. A large fraction of them, especially those inclined to programming, eventually appreciate it. Writing in LaTeX is very similar to writing in a programming language since it involves syntax. In principle, it is possible to write LaTeX in the simplest text editor, like Notepad for Windows. However, there are many editors, that make the LaTeX experience more pleasant by including syntax highlighting and shortcuts. MikTeX, TeXnicCenter, and Texmaker are examples of such editors. They are all free and allow a substantial amount of customizing regarding font families and size, syntax colors, shortcuts, etc.[7] In our experience the best way to learn how to use LaTeX is "learning by copying" – that is, use and progressively tweak somebody else's files. We also use BibTeX, which is a tool and a file format to compile lists of references in conjunction with LaTeX documents. BibTeX allows the user to create and maintain a master database of references, which can be used in different research projects.

3 Organize your Electronic Workspace

For most applied economists the workspace is the computer, so an organized workspace requires a basic folder structure with self-explanatory names that is easy to navigate through. This is very important when several users access the same material. We tend to follow a folder structure like the one in Table 1. This folder

3 See, for example, the Statistical Software Components Archive maintained by Kit Baum at http://fmwww.bc.edu/RePEc/bocode/.
4 See https://www.r-project.org/ and http://www.mathworks.com/matlabcentral/fileexchange/.
5 In the case of Perl, see http://search.cpan.org/dist/CAM-PDF/. For Python, see https://pypi.python.org/pypi/pdfminer/.
6 For example, http://www.ntg.nl/doc/biemesderfer/ltxcrib.pdf contains a summary list of LaTeX commands.
7 Overleaf is an online LaTeX platform that allows real-time collaboration and fully typeset output. Its Pro and Pro+ versions offer integration with Dropbox.

Table 1: Folder Structure.

Folder	Subfolders
Data	Archive, Input, Temp, Logs, Scripts, Output
Analysis	Archive, Input, Temp, Logs, Scripts, Output, Figures, Tables
Literature	Archive, topic
Background	Archive, topic
Drafts	Archive
Notes	Archive
Communications	Archive, Inbox, Outbox

structure will also serve as the skeleton for the remainder of our discussion, much of which will be a description of our protocols for each folder.

In the Data folder, we keep original, as well as clean data files, along with the corresponding codes that create the clean data files from the original ones. We reserve the Analysis folder for data-related tasks such as summary statistics, graphs, regressions, simulations, etc. For those working with multiple datasets, we suggest a Data folder for each of these datasets. Similarly, we keep separate Analysis folders for each type of analysis.

In the Literature folder, we keep PDF files of the various academic papers we cite in our work. When the number of files increases, it is probably a good idea to organize them by topic, or some other principle. In the Background folder, we have all files with material that is useful background to our research but is not an academic paper, such as industry and regulatory background. This is material that helps us learn about the topic we will be studying as we start a new project and many of them will be cited in the final paper.

In the Drafts folder, we keep a "live" draft, which we continuously update and edit, as well as archived versions of the paper. Archiving is better than deleting because writing for most is a dreadful task and space is cheap. We treat the material in the Notes folder as a long log – most of the times longer than needed – of thoughts, relevant material we have identified, to-do-lists, fruitless conversations, etc. Actually, "kitchen sink" might be a more appropriate name for it. Finally, in the Communications folder, we store communications related to our work, almost exclusively e-mail exchanges.

Before we go into the details of our proposed structure for each of these folders we have a few comments on how to keep control of multiple versions of a particular file and when to delete old/discarded files. Whenever you need to change a file or a script that somebody else has created or written, we suggest you make a copy of the associated file with your initials. Additionally, using a prefix in the file name with the date written as "yyyy.mm.dd.", such as `2015.07.10_TASK_KM.do`, may also be helpful. Over time, getting into a purging mode to clean outdated files is a healthy exercise. Again, since space is cheap these days, it is probably better to archive than delete material until the project has reached a particular milestone or is close to completion. You can delete files once you decide that they contain material that is not needed anymore. Until you reach this purging stage, it is a good practice to maintain an Archive folder for old versions of files that are updated and at some point will be eliminated. Prefixing the Archive folder with a character such as the underscore or the tilde will make it more visible since it will (almost always) appear in the top of your screen.

4 Data

4.1 Organizational Structure

Empirical projects typically rely on more than one dataset. For each dataset, we recommend the following folder structure: Input, Temp, Logs, Scripts, and Output. We keep the raw data in the Input folder and store files created during intermediate steps of data processing in the Temp folder. The clean datasets go in the

Output folder. It is important to note that, in order to be able to replicate the analysis, we keep the original data files in their raw state. Any processing of the data will be saved in a separate file. We write code for the vast majority of the data processing; we keep these code files in the Scripts folder. There are, at least, two very good reasons for writing scripts rather than modifying data in an excel file or by running commands in Stata without saving them in a script: you document what you did and, most importantly, you, your collaborators, or somebody else, can replicate and check what you did.

Maintaining a log file for each data-processing script is a good habit to develop. It gives the user the chance to track various steps of the data processing and identify issues such as duplicate records and missing values of data fields. It also allows you to review the processed data without having to re-run potentially time-consuming codes. Keeping such files in the Logs folder and using the same name as the data-processing script to ease cross checks is our way to go here. We store the processed data in a format makes it ready to be analyzed in the Output folder. We view all these steps as our preliminary triage of the data.

4.2 Documentation

The basic principle for documenting a dataset is that the documentation should be informative enough for anybody to retrieve the same data without any assistance. In addition, documentation is very useful for citations and easy checks for data updates and revisions, which are very common for data released by government agencies, for example.

One convenient way to document the data sources is to have a ReadMe file in the Input folder. When it comes to data accessed from the Web, we suggest providing a detailed URL and the date the URL was accessed. For example, citing www.eia.gov is not a good way of citing a particular dataset from the US Energy Information Administration since there is an enormous amount of data in the agency's website – common sense should always come into play. A simple ReadMe text file works well most of the time. When working with others, it is a good idea to indicate the person that collected the data (e.g. collected by John Doe on Jan-25-2015); assuming responsibility in this way saves time and back-and-forth when in doubt about the origin of the data.

An Excel or Word file can be used to store screenshots of websites taken while downloading data by using `CTRL+Print Screen` and `CTRL+V`, to track various steps taken to access the data. Again, common sense should come into play when it comes to this process. Although it seems to be a hassle, it can avoid wasting time in the future when trying to recreate the data downloaded or when updating the data when a new version becomes available. For a large number of files accessed from the Web, we recommend using Perl or Python.

4.3 Scripts

Our suggestions in this section are based exclusively on our experience with Stata, which is very popular among economists. However, based on conversations with colleagues and students, we feel confident that analogous suggestions are valid for other popular statistical packages. Again, we do not intend to identify the "best" package with which to do data work but simply provide what, in our experience, are useful standards for coding. Here is a list of things to keep in mind when working with Stata:

1. Learn to read the manual. Stata has a very nice PDF-based documentation of all its commands. Typing `man command` or `help command` (e.g. `man regress`) in its command line will display a description of the command including hyperlinks that transition to the PDF-based environment, which is more pleasant to read from. More importantly, the PDF documentation contains a section on *Methods and Formulas* about the various estimation routines available.
2. Write code in a consistent manner. Essentially, develop a code identity. This is very helpful among collaborators when they share and review codes. Some useful and standard procedures include: define all

paths where various files are at the very beginning of the script, then import data, then proceed with manipulating the data, etc.; and do it in a consistent manner. It is very helpful to get familiar with somebody else's style early on and then develop your own.

For example, it is standard to place the following commands at the top of the script:

(a) `clear all`: clears the memory
(b) `capture log close`: closes any lingering log files
(c) `set type double`: uses double numerical precision when you import files;
(d) `set more off`: avoids the need to hit any key for multiple screens of code to be executed. Recent versions of Stata do not require memory settings to load files, so there is no need to use `set memory` as it was the case for previous versions

3. Use globals or locals to define (ideally, relevant) paths where files are kept at the beginning of the scripts. There is probably a better way to say this: "Do not hardcode paths", especially, if you use them repeatedly, and here is why. Imagine that you clean or rename folders or your co-author wants to use the same scripts but his Dropbox folders have a different structure. Changing a global in the beginning of the script is very close to trivial while changing multiple lines of hardcoded paths is *casus belli*.
4. Because the Stata editor does not automatically break lines to fit the screen size, do not write commands that exceed a reasonable number of columns – 80 is a good number to have in mind even when you use a 24-inch screen, as is very common these days. If you want to use long commands, which is often the case with formatting options in figures, use `#delimit ;` which allows you to split a single command into multiple lines.
5. Use many, but meaningful, comments in your code that can be used to identify major steps in what the code does and most importantly remind you why you did something when you did it a long time ago. A couple of sentences explaining what the code does at the top is always helpful. Make your comments in the body of the code easy to spot. For example, never make them longer than 80 characters – as mentioned above, any monitor can accommodate these number of characters without having to scroll left and right. Place a line of the same length above and below your comments with just asterisks (or any symbol you like) that will serve as dividers. We have found that this makes comments easy to spot.
6. Many times there is a need to import and process multiple data files with identical structure. For example, in one of our research projects we had to import into Stata several comma-delimited (CSV) files from the US Environmental Protection Agency that contained emissions-related information in a separate file for each state and month. In these situations, we recommend the use of loops. In this case, one loop would be over states, the second would be over years, and the last one would be over months. Use the locals from the loops to name the files. When you use loops, make sure you indent your code. Unfortunately, the Stata editor does not have automatic indentation. Assuming that the locals `i`,`j`, and `k` are used for the three nested loops, you can then save a file with the name `emissions_`i'_`j'_`k'`.dta in the Temp folder. Since the file names are now standardized, you can append these intermediate files using loops in the same spirit and save a file, say, with the name `emissions_all.dta` in the Output folder. Other than elegance, there is one more good reason for using loops as opposed to "copying-and-pasting" – most errors in coding happen during copying and pasting and not properly updating!
7. Get used to structuring the code in "IMPORT-and-APPEND" steps that are distinct and can be isolated if necessary. For example, it should be possible to comment the code in the IMPORT step (i.e. transform the code into a comment so it does not perform the importing commands), which can be quite time-consuming, while editing and running code for the APPEND step. Compartmentalizing is a very desirable feature of anyone's code.
8. It is better to maintain original variable names when you import your data – in some cases, you may have to change them if the variable names exceed the length that Stata can accommodate. You should also check your data for duplicate records, which you should flag using say an indicator variable, which you may name `flag_dup_tag`.
9. When you generate variables that contain similar information, it is helpful to use names such that you can easily order, summarize and edit using wildcards. For example, assume that you a have a

nominal-price and a real-price variable. It probably makes sense to name them `price_nom` and `price_real` (instead of `nom_price` and `real_price`) because you can then type `sum price_*` and get their summary statistics.

10. Avoid using long variable names; you can always find meaningful short names. For example, `gross_domestic_product_united_states` is not much (if any) more informative than `gdp_usa`.
11. Learn how to use regular expressions (`regexm`). They are extremely powerful in cleaning and standardizing string fields. You can also use them to filter the data using string fields.
12. Learn how to properly merge data. We highly recommend to use the one-to-one (1:1), one-to-many (1:m), and many-to-one (m:1) options. It is also a good idea to use an `assert` after the merging to ensure that your merge did what it was supposed to do. Merging incorrectly is a common and expensive coding error that often goes unnoticed.
13. Learn how to reshape data from long to wide and vice versa. This is done using the `reshape` command. One reason to get familiar with the reshaping is the Stata-Excel interaction. Excel pivot tables work best with data in long format.
14. Format your numeric variables when you store them. For example, the comma format in Stata `format%XX.YYf` allows to dictate YY decimals from a total of XX digits. Numbers larger than a billion, are hard to track, so use millions or thousands. Formatting does not affect the accuracy of how the numeric variable is stored but it helps the eye. When you export data from Stata to Excel make sure you carry a large number of decimal places. Learn also the alternative ways to format date-related fields and pay attention on how you export dates. You need to format date fields in Stata in a specific way so that they are not distorted when you import the data from or export to Excel.
15. Be particularly careful about handling missing values when using the `collapse` and `egen` commands, which are often used to generate summary statistics. For example, if the variable `quantity` has only missing values, the command `collapse (mean) quantity` will generate "." values, which is how Stata stores missing values, but the command `collapse (sum) quantity` will produce zeroes. The `egen` command behaves in the same way.
16. Learn how to extract information from Stata output using `return list` and `ereturn list`. It is very handy when you want to add information in the bottom of a table with estimation results, say using `outreg2`.
17. You can highlight segments of code over multiple lines and execute them by hitting `ctrl+D` or `ctrl+R`. The former will produce output in the screen, while the latter will not. For those also familiar with MATLAB, it is equivalent to selecting code and hitting `F9` and it can be very handy as you try things on the fly.

As a final bonus trick, note that it is very simple to have Stata send you an e-mail once a script is completed in Windows systems. This can come very handy when you have scripts in the background running for several hours or days thanks to the `emailme` routine.[8]

4.3.1 Debugging

Talking about coding always leads to the issue of debugging. Debugging is like flossing. It is very good for you, rather annoying to do, and can have serious implications if it is not done properly or worst case scenario, at all. It is a very ungrateful task and there are not many willing do it for you unless they have some "skin in the game," such as research assistants and co-authors, or others that are just looking for a mistake in somebody's code. The bad thing about coding errors is that in many cases they do not generate obvious red flags and, therefore, are hard to catch.

8 https://ideas.repec.org/c/boc/bocode/s457879.html.

Based on our experience, the first best is to have two different individuals writing different scripts performing the same task and then compare their answers. If their answers are the same then they are either both right or they have both made the same mistakes; this latter outcome is highly unlikely. Of course, having two individuals writing code that performs the same task from scratch, may not be possible for a variety of reasons.

There are two alternatives. The second best is to ask someone to review the code. The third best is to review the code yourself after a period of time. Even a day or two make a lot of difference in catching coding errors. If you do that you will be surprised how many times you will find yourself wondering: "Did I really write this?"

5 Analysis

At this point, you have processed the data and they are ready to be analyzed to answer the hard and policy-relevant questions using a robust identification approach. We use the Analysis folder as a repository of the core empirical work of a project. As mentioned earlier, we suggest you use different subfolders for each type of analysis. Examples include Summary Statistics, Reduced-form Analysis, Counterfactuals, Robustness. We tend to use the folder structure in Table 1 for each of them. Below we offer suggestions and tips for creating tables and charts, which you will surely need for any kind of analysis.

5.1 Tables and Figures using Excel and LaTeX

If you prefer the format of Excel figures and tables rather than those produced by Stata, you can use Stata to prepare the data that will be used in those figures and tables and then export them to Excel using `export excel`. Examples include tabulations, summary statistics, estimation output (e.g. regression maximum likelihood, etc.). Once you have exported the data to Excel, you can format figures or tables to your liking and then create PDF files using the SAVE-AS-PDF option. If you need to repeat this process many times or to simplify the updating process, you can write a simple Excel macro that replicates the SAVE-AS-PDF process. However, to date, Stata does not allow exporting to macro-enabled Excel files but there is a way around this problem, which we describe in an example below.

Once you generate a PDF file associated with a table/figure, you can import it in a LaTeX document using `includegraphics` which also allows you to crop – eliminate unnecessary blank spaces – and resize the image of the table/figure. Of course you cannot edit the fonts, axes' labels, captions, etc., since the underlying file is an image file. Following these steps, you have fully automated a pretty lengthy and very often repetitive process. The process is also "copy-and-paste" error- and hassle-free since it avoids moving numbers around by hand. In addition, creating and formatting tables in LaTeX is pretty time consuming. With the process we just described you can do all the table formatting in Excel, which is much easier than doing it in LaTeX.

Here is an example of how the whole process works.

1. Export your data from Stata to an XLSX file.
2. Prepare a macro-enabled Excel file (.XLSM) which you link to the XLSX file – to date, Stata cannot export to macro-enabled Excel files. You will export your data to the XLSX file which will automatically update the linked XLSM file.
3. Create two tabs in the XLSM file. Use the first tab in the XLSM to link to the data in the xlsx file; let's call it DATA. Use the second tab, say TABLE, to create a formatted table whose entries are linked to the cells of the DATA tab.
4. Create a macro in the XLSM file that saves the TABLE tab as a PDF file. If you do not want or need to use the the automated process to create PDF version of the table, you do not need to create the XLSM file and can simply work with a single XLSX file.

Once you do this, each time you generate new data in Stata and you export them in Excel your formatted nice-looking table in TABLE tab is updated. You can also create another tab, say FIGURE, which this time produces a figure that is also automatically updated. You can follow the same approach in combination with the `outreg2` command in Stata that allows you to save XML files with tables of regression estimation results. Additional commands that perform tasks similar are also available as Stata add-ins.

Here is one more trick. If you use the custom format "['0.000']" in Excel, it will format the numbers in a cell so that they have three decimal points and are surrounded by squared brackets, which you can use for the *p*-value in a table of estimation results. Similarly, you can use "('0.000')" to have numbers with three decimal points surrounded by parentheses, which you can use for standard errors. The LM Roman OTF family of fonts in Excel generates fonts that are almost indistinguishable from the CMR family in LaTeX. Unfortunately, these fonts do not work with the SAVE-AS-PDF option in Excel and, as a result, with any macros implementing it.[9]

5.2 Stata Figures

Stata has greatly improved its graphics features so you might decide to skip the step of creating figures in Excel and do them directly in Stata. Take the time to create a template code of a few basic graphs with your preferred formatting options so that you can reuse it with a minimum amount of adjustments. Examples include axes titles and labels, captions, plot regions, etc. Be sensitive to the amount of information you put in a figure and avoid the clutterplots – see the excellent article by Schwabish (2014) on how to avoid major pitfalls.[10]

When you are in the data exploration stage you will typically want to see your data in multiple figures. Get used to writing loops that will help you generate a large number of figures, such as scatter plots and time series plots that can be used to get familiar with your data. For example, you can write a loop in Stata that saves figures in a PNG format, save them in a folder and then navigate through them in seconds just as you would when browsing through photos. Alternatively, if you save the figures in PDF format, you can then import them in seconds in a LaTeX document. Once imported to the LaTeX document, they are also updated automatically every time changes take place. In the same document, you can write comments for a large number of figures that will help you better understand your data and recall your preliminary observations of the data quickly.[11]

5.3 Pivot Tables and Charts in Excel

If you prefer Excel's charts, the Pivot Tables and Charts tools provide a very convenient way to slice the data in many ways using various filters on the fly. This also means you have to be very careful. Essentially, this feature allows to – with a single click and drag – add/remove variables in a table/figure, change the axis the

9 We also recommend the readers to familiarize themselves with the Adobe Acrobat "Export Selection As" capabilities, which allow the user to export tables from a PDF file in Excel or CSV files. We have used this method extensively to extract tables from various EIA publications. The same approach can be used to extract figures. See also the 10 commandments for Regression Tables by Keith Head at http://strategy.sauder.ubc.ca/head/tabcoms.htm for useful tips.

10 We learnt how to make graphs in Stata tweaking many of the examples available at http://www.ats.ucla.edu/stat/stata/library/GraphExamples/. See also the 10 commandments for Figures by Keith Head at http://strategy.sauder.ubc.ca/head/figcoms.htm for useful tips.

11 The use of maps, especially for those doing empirical research fields, such as agricultural economics, development, trade, and energy, to name a few examples, is very common these days. Although creating maps in Stata is possible, using the `spmap` command, the final product is not as appealing as the one generated by programs, such as Tableau, Cartodb, or Mapbox, because Stata is not a mapping software per se. The ESRI ArcGIS is considered by many the gold standard when it comes to mapping.

data are displayed on, change the chart type, etc. The shortcut for the Pivot Table tool is `ALT->D->P;` once you select the data that you want to use in a pivot table, your Pivot Table is ready. If you try to create a chart within a Pivot Table, you essentially create a Pivot Chart.

A very powerful feature of the Pivot Tables is the calculated field. Using a very simple example, which most probably does not do justice to the feature's capabilities, you can calculate price as total revenue divided by quantity. Using a pivot table, you can perform simple tasks such as calculating market shares or prices across various cross-sectional and time dimensions, such as by city and year, or by state and year, or by state, or by year.

6 Communications

We use an Inbox and an Outbox folder to save any kind of communications; typically emails. In many cases, you will contact someone that will be the "Person Most Knowledgeable" on a topic that is relevant to your empirical analysis. Examples of such topics include data collection methods, industry background, regulatory developments. Very often, these individuals do you a favor when responding to your e-mails, so keep that in mind when you draft the e-mail. Never forget to send a "Thank-You" e-mail.

Here is another useful tip. Many of the data subscription services come with e-mail news alerts. For example, our subscription to SNL Energy included e-mail alerts regarding regulatory developments in the US electric power sector. Assuming that you use Outlook, which many of us do, it is probably a good idea to create a rule so that e-mails from your data subscription services go to a particular place in your Inbox. Periodically, you can export the msg Outlook files or their attachments to your Inbox folder and you can do using VBA scripts.[12] You can then cite these communications in the paper (e.g. e-mail communication of May 15, 2015 with John Doe) or simply keep them for documentation purposes. At the same time, this will allow you to acknowledge various individuals in your work, which is a nice thing to do.

7 Literature

We use the Literature folder to collect all the papers that we reviewed and will cite in the research paper. With the risk of stating the obvious, we suggest you save the papers in PDF format and name the files using a convention. Our preferred option is: `author_source_year.pdf`. For example, `Someone_and_Other_ReStat_2014.pdf`. For more than two authors, we recommend using et al. as in `Someone_et_al_AER_1999.pdf`.

It is also useful to learn how to use BibTeX for LaTeX, which provides a very convenient way to organize your references. Most importantly, saves you the hassle of formatting the Reference section of your paper. In addition, many databases for accessing academic journals, with JSTOR being an example, allow you to download BibTeX entries for the articles you are accessing. A Java-based application named `JabRef` allows you to open, edit, and save BibTeX files in a database-spreadsheet environment. It is very simple to use and it is free. A BibTex file gives you the ability to assign keys to the various cites, which you can then use very conveniently in LaTeX along with the `citet` command. You should agree on a naming convention for the BibTex key with your collaborators; most of the times `Aaayyyy`, `AaaBbbbyyyy`, and `Aaaetalyyyy` conventions generate enough keys, so that there is no overlap even if you are building a master database of several hundreds of references. For example, you can use `Som2012` as a key for Someone (2012), `SomOth2014` for Someone and Other (2014) and `Someta1999` as a key for Someone et al. (1999). Establishing these conventions early on, especially when working with others, saves a lot of time.

12 For an example on how to export attachments, see http://www.rondebruin.nl/win/s1/outlook/saveatt.htm.

Alternatively, we have found it helpful to have a spreadsheet, say `Index.xlsx`, which can be created very easily with Python, with three columns: file name, file link, notes.[13] Even if you do not know Python you can create such a spreadsheet very easily using the `hyperlink` function in Excel. In the notes column, you can have a very short description of the paper. Additional columns that indicate the relevance of the paper using a point system and the topic (e.g. Cap and Trade) can be very helpful for review purposes. Several colleagues and students utilize reference management software, like Mendeley or Zotero, which also allow you to organize Web searches in a very efficient way. These are pretty helpful things to do when you initially collect material and once you get used to creating them they take almost no time and keep you very well organized.

8 Background and Notes

We use the Background folder to save material related to industry background, legislation, etc. As in the case of the Literature folder, it is recommended to create an `Index.xlsx` file with links to PDF files and a brief comment what each file is about.

Use the Notes folder to keep notes of the material that you have been reviewing and notes regarding conclusions you reach as you proceed with your analyses. Not all analyses you perform will end up in the final research paper so it is useful to keep track of what was tried and why it was discarded. Keeping a single log file of these notes will make it easy to search through if you need to recall facts, conclusions or questions.

9 Drafts

As we mentioned earlier, we prefer to write our papers in LaTeX. First of all, it offers high-quality professional typesetting and it allows you to switch from text mode to math mode very easily by enclosing anything that needs to be in math mode between $ signs. For example, $y=a+bx$ will generate $y=a+bx$. A distinct feature of every LaTeX file is its preamble, which allows you to include several packages to enhance its functionality; the list of packages available is enormous. Here is a short list of things you can do using LaTeX very easily

1. Anything preceded by % is treated as comment. This is extremely useful to add notes in your `.tex` file – where you write LaTeX – that do not appear in the final document.
2. Create sections using `section{}`, `subsection{}`, etc.
3. Create cross-references using `ref{}` and `label{}`
4. Add citations using `citet{}` along with a BibTeX file
5. Import and edit figures using `includegraphics[]{}`
6. Include URL links using `url{}`
7. Start at the top of a new page using `clearpage`
8. Highlight text, which helps for editing, using `hl{}` from the `soul` package

Version control is a fine balancing act and one way to do it is to add suffixes to the file names such as `v01`, `v02`, etc. to indicate versions. As we mentioned before, we prefer to archive than to delete since writing is a dreadful task and the size of `tex` files is negligible. You can use the Archive folder to stay organized. We also use extensively the comment capability of LaTeX to include things such as links to the folders that contain the analysis in a particular section, or suggestions as the draft goes through multiple rounds of editing.

As final pieces of advice, number your equations, to the extent possible, and use captions in tables and figures even for preliminary drafts. It is much easier to communicate edits by referring to, say equation (10), as opposed to "the equation that is in the middle of page 10," especially when there are multiple equations

13 https://docs.python.org/2/library/os.html.

on the same page. Let a number of days go by before you begin iterating on editing the draft again and, most importantly, give the draft to someone who has not seen it before. In addition, have a second screen, keyboard, and mouse on your desk. It is extremely efficient to edit drafts with someone in the same room. If you work with someone long distance, sharing your screen while editing the draft using the share-screen feature in Skype can be very helpful, too. While editing, keep a copy of the celebrated *Chicago Manual of Style* on your desk. It covers a variety of topics from manuscript preparation and publication to grammar, usage, and documentation. It is also available on line.[14]

10 Conclusions

Having spent more than 15 years doing empirical research, we felt the need to write down this set of practices that we have found helpful in doing our work efficiently and with transparency. This set of practices is by no means unique or comprehensive but it has served us, and a large number of colleagues, well over the years. They should be viewed as a set of suggestions on how to access, process, organize, and analyze data efficiently. We hope that young economist employed both in academia and outside of it to find them helpful.

References

Ball, R., and N. Medeiros. 2012. "Teaching Integrity in Empirical Research: A Protocol for Documenting Data Management and Analysis." *The Jounral of Economic Education* 43(2): 182–189.

Cameron, A., and P. Trivedi. 2005. *Microeconometrics Methods and Applications*. Cambridge: University Press.

Gentzkow, M., and J. Shapiro. 2014. "Code and Data for the Social Sciences: A Practitioner's Guide." Working paper, Chicago Booth and NBER.

Schwabish, J. 2014. "An Economist's Guide to Visualizing Data." *Journal of Economic Perspectives* 28(1): 209–234.

Wooldridge, J. 2013. *Introductory Econometrics: A Modern Approach*. 5th ed. South-Western, Cengage Learning.

14 http://www.chicagomanualofstyle.org/home.html.