KRISTOFFER BJÄRKEFUR
LUÍZA CARDOSO DE ANDRADE
BENJAMIN DANIELS
MARIA JONES

DATA FOR DEVELOPMENT IMPACT: THE DIME ANALYTICS RESOURCE GUIDE

DIME ANALYTICS

PUBLISHED BY DIME ANALYTICS

http://worldbank.github.com/d4di

First printing, January 2020

Notes on this edition

This is a draft peer review edition of *Data for Development Impact: The DIME Analytics Resource Guide*. This version of the book has been substantially revised since the first release in June 2019 with feedback from readers and other experts. It now contains most of the major content that we hope to include in the finished version, and we are in the process of making final additions and polishing the materials to formally publish it.

This book is intended to remain a living product that is written and maintained in the open. The raw code and edit history are online at: `https://github.com/worldbank/d4di`. You can get a PDF copy at: `https://worldbank.github.com/d4di`. The website also includes the most updated instructions for providing feedback, as well as a log of errata and updates that have been made to the content.

Feedback

Whether you are a DIME team member or you work for the World Bank or another organization or university, we ask that you read the contents of this book carefully and critically. We encourage feedback and corrections so that we can improve the contents of the book in future editions. Please visit `https://worldbank.github.com/d4di/feedback/` to see different options on how to provide feedback. You can also email us at `dimeanalytics@worldbank.org` with input or comments, and we will be very thankful. We hope you enjoy the book!

Abbreviations

2SLS – Two-Stage Least Squares
AEA – American Economic Association
CAPI – Computer-Assisted Personal Interviewing
CI – Confidence Interval
DEC – Development Economics Group at the World Bank
DD or DiD – Differences-in-Differences
DGP – Data-Generating Process
DIME – Development Impact Evaluations
FC – Field Coordinator
FE – Fixed Effects
HFC – High-Frequency Checks
IRB – Instituional Review Board
IV – Instrumental Variables
MDE – Minimum Detectable Effect
NGO – Non-Governmental Organization
ODK – Open Data Kit
OLS – Ordinary Least Squares
OSF – Open Science Foundation
PI – Principal Investigator
PII – Personally-Identifying Information
QA – Quality Assurance
RA – Research Assistant
RD – Regression Discontinuity
RCT – Randomized Control Trial
SSC – Statistical Software Components
WBG – World Bank Group

Contents

Dedicated to all the research assistants
who have wrangled data without being
taught how, hustled to get projects done
on time, wondered if they really should get
their PhD after all, and in doing so made
this knowledge necessary and possible.

Introduction: Data for development impact

Welcome to Data for Development Impact. This book is intended to serve as a resource guide for people who collect or use data for development research. In particular, the book is intended to guide the reader through the process of research using primary survey data, from research design to fieldwork to data management to analysis. This book will not teach you econometrics or epidemiology or agribusiness. This book will not teach you how to design an impact evaluation. This book will not teach you how to do data analysis, or how to code. There are lots of really good resources out there for all of these things, and they are much better than what we would be able to squeeze into this book.

What this book will teach you is how to think about quantitative data, keeping in mind that you are not going to be the only person collecting it, using it, or looking back on it. We hope to provide you two key tools by the time you finish this book. First, we want you to form a mental model of data collection as a "social process", in which many people need to have the same idea about what is to be done, and when and where and by whom, so that they can collaborate effectively on large, long-term projects. Second, we want to provide a map of concrete resources for supporting these processes in practice. As research teams and timespans have grown dramatically over the last decade, it has become inefficient for everyone to have their own personal style dictating how they use different functions, how they store data, and how they write code.

Doing credible research at scale

The team responsible for this book is known as **DIME Analytics**.[1] The DIME Analytics team works within the **Development Impact Evaluation (DIME)** group [2] at the World Bank's **Development Economics group (DEC)**.[3] After years of supporting hundreds of projects and staff in total, DIME Analytics has built up a set of ideas, practices, and software tools that support all of the research projects operated at DIME.

In the time that we have been working in the development field, the proportion of projects that rely on **primary data** has soared.[4] Today, the scope and scale of those projects continue to expand rapidly. More and more people are working on the same data over longer timeframes. This is because, while administrative datasets and **big data** have important uses, primary data[5] is critical for answering specific research questions.[6] As the ambition of development researchers grows, so too has the complexity of the data on which

[1] http://www.worldbank.org/en/research/dime/data-and-analytics

[2] http://www.worldbank.org/en/research/dime

[3] http://www.worldbank.org/en/research/

[4] Angrist, J., Azoulay, P., Ellison, G., Hill, R., and Lu, S. F. (2017). Economic research evolves: Fields and styles. *American Economic Review*, 107(5):293–97

[5] **Primary data:** data collected from first-hand sources.

[6] Levitt, S. D. and List, J. A. (2009). Field experiments in economics: The past, the present, and the future. *European Economic Review*, 53(1):1–18

they rely to make policy-relevant research conclusions from **field experiments**.[7] Unfortunately, this seems to have happened (so far) without the creation of guidelines for practitioners to handle data efficiently and transparently, which could provide relevant and objective quality markers for research consumers.

[7] **Field experiment:** experimental intervention in the real world, rather than in a laboratory.

One important lesson we have learned from doing field work over this time is that the most overlooked parts of primary data work are reproducibility and collaboration. You may be working with people who have very different skillsets and mindsets than you, from and in a variety of cultures and contexts, and you will have to adopt workflows that everyone can agree upon, and that save time and hassle on every project. This is not easy. But for some reason, the people who agreed to write this book enjoy doing it. (In part this is because it has saved ourselves a lot of time and effort.) As we have worked with more and more DIME recruits we have realized that we barely have the time to give everyone the attention they deserve. This book itself is therefore intended to be a vehicle to document our experiences and share it with with future DIME team members.

The **DIME Wiki** is one of our flagship resources designed for teams engaged in impact evaluation projects. It is available as a free online collection of our resources and best practices.[8] This book therefore complements the detailed-but-unstructured DIME Wiki with a guided tour of the major tasks that make up primary data collection.[9] We will not give a lot of highly specific details in this text, but we will point you to where they can be found, and give you a sense of what you need to find next. Each chapter will focus on one task, and give a primarily narrative account of: what you will be doing; where in the workflow this task falls; when it should be done; who you will be working with; why this task is important; and how to implement the most basic form of the task.

[8] http://dimewiki.worldbank.org/

[9] Like this: https://dimewiki.worldbank.org/wiki/Primary_Data_Collection

We will use broad terminology throughout this book to refer to different team members: **principal investigators (PIs)** who are responsible for the overall success of the project; **field coordinators (FCs)** who are responsible for the operation of the project on the ground; and **research assistants (RAs)** who are responsible for handling technical capacity and analytical tasks.

Writing reproducible code in a collaborative environment

Research reproduciblity and data quality follow naturally from good code and standardized practices. Process standardization means that there is little ambiguity about how something ought to be done, and therefore the tools that are used to do it are set in advance. Good code is easier to read and replicate, making it easier to spot mistakes.

The resulting data contains substantially less noise that is due to sampling, randomization, and cleaning errors. And all data work can be easily reviewed before it's published and replicated afterwards.

A good do-file consists of code that has two elements: - it is correct (doesn't produce any errors along the way) - it is useful and comprehensible to someone who hasn't seen it before (such that the person who wrote this code isn't lost if they see it three weeks later) Most research assistants that join our unit have only been trained in how to code correctly. While correct results are extremely important, we usually tell our new research assistants that *when your code runs on your computer and you get the correct results then you are only half-done writing good code.*

Just as data collection and management processes have become more social and collaborative, code processes have as well.[10] This means other people need to be able to read your code. Not only are things like documentation and commenting important, but code should be readable in the sense that others can: (1) quickly understand what a portion of code is supposed to be doing; (2) evaluate whether or not it does that thing correctly; and (3) modify it efficiently either to test alternative hypotheses or to adapt into their own work.[11]

[10] https://dimewiki.worldbank.org/wiki/Stata_Coding_Practices

[11] https://kbroman.org/Tools4RR/assets/lectures/07_clearcode.pdf

To accomplish that, you should think of code in terms of three major elements: **structure**, **syntax**, and **style**. We always tell people to "code as if a stranger would read it", from tomorrow, that stranger will be you. The **structure** is the environment your code lives in: good structure means that it is easy to find individual pieces of code that correspond to tasks. Good structure also means that functional blocks are sufficiently independent from each other that they can be shuffled around, repurposed, and even deleted without damaging other portions. The **syntax** is the literal language of your code. Good syntax means that your code is readable in terms of how its mechanics implement ideas – it should not require arcane reverse-engineering to figure out what a code chunk is trying to do. **Style**, finally, is the way that the non-functional elements of your code convey its purpose. Elements like spacing, indentation, and naming (or lack thereof) can make your code much more (or much less) accessible to someone who is reading it for the first time and needs to understand it quickly and correctly.

For some implementation portions where precise code is particularly important, we will provide minimal code examples either in the book or on the DIME Wiki. In the book, they will be presented like the following:

code.do

```
* Load the auto dataset
    sysuse auto.dta , clear

* Run a simple regression
    reg price mpg rep78 headroom , coefl

* Transpose and store the output
    matrix results = r(table)'

* Load the results into memory
    clear
    svmat results , n(col)
```

We have tried really hard to make sure that all the Stata code runs, and that each block is well-formatted and uses built-in functions. We will also point to user-written functions when they provide important tools. In particular, we have written two suites of Stata commands, `ietoolkit`[12] and `iefieldkit`,[13] that standardize some of our core data collection workflows. Providing some standardization to Stata code style is also a goal of this team, since groups are collaborating on code in Stata more than ever before. We will not explain Stata commands unless the behavior we are exploiting is outside the usual expectation of its functionality; we will comment the code generously (as you should), but you should reference Stata help-files `h [command]` whenever you do not understand the functionality that is being used. We hope that these snippets will provide a foundation for your code style. Alongside the collaborative view of data that we outlined above, good code practices are a core part of the new data science of development research. Code today is no longer a means to an end (such as a paper), but it is part of the output itself: it is a means for communicating how something was done, in a world where the credibility and transparency of data cleaning and analysis is increasingly important.

[12] https://dimewiki.worldbank.org/wiki/ietoolkit

[13] https://dimewiki.worldbank.org/wiki/iefieldkit

While adopting the workflows and mindsets described in this book requires an up-front cost, it should start to save yourself and others a lot of time and hassle very quickly. In part this is because you will learn how to do the essential things directly; in part this is because you will find tools for the more advanced things; and in part this is because you will have the mindset to doing everything else in a high-quality way. We hope you will find this book helpful for accomplishing all of the above, and that you will find that mastery of data helps you make an impact!

– The DIME Analytics Team

Chapter 1: Handling data ethically

Development research does not just *involve* real people – it also *affects* real people. Policy decisions are made every day using the results of briefs and studies, and these can have wide-reaching consequences on the lives of millions. As the range and importance of the policy-relevant questions asked by development researchers grow, so does the (rightful) scrutiny under which methods and results are placed. Additionally, research also involves looking deeply into real people's personal lives, financial conditions, and other sensitive subjects. The rights and responsibilities involved in having such access to personal information are a core responsibility of collecting personal data. Ethical scrutiny involves two major components: **data handling** and **research transparency**. Performing at a high standard in both means that consumers of research can have confidence in its conclusions, and that research participants are appropriately protected. What we call ethical standards in this chapter are a set of practices for research quality and data management that address these two principles.

Neither transparency nor privacy is an "all-or-nothing" objective. We expect that teams will do as much as they can to make their work conform to modern practices of credibility, transparency, and reproducibility. Similarly, we expect that teams will ensure the privacy of participants in research by intelligently assessing and proactively averting risks they might face. We also expect teams will report what they have and have not done in order to provide objective measures of a research product's performance in both. Otherwise, reputation is the primary signal for the quality of evidence, and two failures may occur: low-quality studies from reputable sources may be used as evidence when in fact they don't warrant it, and high-quality studies from sources without an international reputation may be ignored. Both these outcomes reduce the quality of evidence overall. Even more importantly, the only way to determine credibility without transparency is to judge research solely based on where it is done and by whom, which concentrates credibility at better-known international institutions and global universities, at the expense of quality research done by people and organizations directly involved in and affected by it. Simple transparency standards mean that it is easier to judge research quality, and making high-quality research identifiable also increases its impact. This section provides some basic guidelines and resources for using field data ethically and responsibly to publish research findings.

Protecting confidence in development research

Across the social sciences, the open science movement has been fueled by discoveries of low-quality research practices, data and code that are inaccessible to the public, analytical errors in major research papers, and in some cases even outright fraud. While the development research community has not yet experienced any major

scandals, it has become clear that there are necessary incremental improvements in the way that code and data are handled as part of research. Major publishers and funders, most notably the American Economic Association, have taken steps to require that these research components are accurately reported and preserved as outputs in themselves.[14]

The empirical revolution in development research has therefore led to increased public scrutiny of the reliability of research.[15] Three major components make up this scrutiny: **reproducibility**[16], **transparency**,[17] and **credibility**.[18] Development researchers should take these concerns seriously. Many development research projects are purpose-built to address specific questions, and often use unique data or small samples. This approach opens the door to working closely with the broader development community to answer specific programmatic questions and general research inquiries. However, almost by definition, primary data that researchers use for such studies has never been reviewed by anyone else, so it is hard for others to verify that it was collected, handled, and analyzed appropriately.[19] Maintaining credibility in research via transparent and reproducibile methods is key for researchers to avoid serious errors. This is even more important in research using primary data, and therefore these are not byproducts but core components of research output.

[14] https://www.aeaweb.org/journals/policies/data-code/

[15] Rogers, A. (2017). The dismal science remains dismal, say scientists. *Wired*

[16] Duvendack, M., Palmer-Jones, R., and Reed, W. R. (2017). What is meant by "replication" and why does it encounter resistance in economics? *American Economic Review*, 107(5):46–51

[17] Christensen, G. and Miguel, E. (2018). Transparency, reproducibility, and the credibility of economics research. *Journal of Economic Literature*, 56(3):920–80

[18] Ioannidis, J. P., Stanley, T. D., and Doucouliagos, H. (2017). The power of bias in economics research. *The Economic Journal*

[19] https://blogs.worldbank.org/impactevaluations/what-development-economists-talk-about-when-they-talk-about-reproducibility

Research reproducibility

Reproducible research means that the actual analytical processes you used are executable by others.[20] (We use "reproducibility" to refer to the precise analytical code in a specific study.[21]) All your code files involving data cleaning, construction and analysis should be public, unless they contain identifying information. Nobody should have to guess what exactly comprises a given index, or what controls are included in your main regression, or whether or not you clustered standard errors correctly. That is, as a purely technical matter, nobody should have to "just trust you", nor should they have to bother you to find out what happens if any or all of these things were to be done slightly differently.[22] Letting people play around with your data and code is a great way to have new questions asked and answered based on the valuable work you have already done.[23] Services that log your research process are valuable resources here – GitHub is one of many that can do so.[24] Such services can show things like modifications made in response to referee comments, by having tagged version histories at each major revision. These services can also use issue trackers and abandoned work branches

[20] Dafoe, A. (2014). Science deserves better: the imperative to share complete replication files. *PS: Political Science & Politics*, 47(1):60–66

[21] http://datacolada.org/76

[22] Simmons, J. P., Nelson, L. D., and Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11):1359–1366; Simonsohn, U., Simmons, J. P., and Nelson, L. D. (2015). Specification curve: Descriptive and inferential statistics on all reasonable specifications. *Available at SSRN 2694998*; and Wicherts, J. M., Veldkamp, C. L., Augusteijn, H. E., Bakker, M., Van Aert, R., and Van Assen, M. A. (2016). Degrees of freedom in planning, running, analyzing, and reporting psychological studies: A checklist to avoid p-hacking. *Frontiers in Psychology*, 7:1832

[23] https://blogs.worldbank.org/opendata/making-analytics-reusable

to document the research paths and questions you may have tried to answer as a resource to others who have similar questions.

Secondly, reproducible research[25] enables other researchers to re-use your code and processes to do their own work more easily and effectively in the future. This may mean applying your techniques to their data or implementing a similar structure in a different context. As a pure public good, this is nearly costless. The useful tools and standards you create will have high value to others. If you are personally or professionally motivated by citations, producing these kinds of resources can lead to that as well. Therefore, your code should be written neatly with clear instructions and published openly. It should be easy to read and understand in terms of structure, style, and syntax. Finally, the corresponding dataset should be openly accessible unless for legal or ethical reasons it cannot be.[26]

[25] https://dimewiki.worldbank.org/wiki/Reproducible_Research

[26] https://dimewiki.worldbank.org/wiki/Publishing_Data

Research transparency

Transparent research will expose not only the code, but all the other research processes involved in developing the analytical approach.[27] This means that readers be able to judge for themselves if the research was done well and the decision-making process was sound. If the research is well-structured, and all of the relevant documentation[28] is shared, this makes it easy for the reader to understand the analysis later. Expecting process transparency is also an incentive for researchers to make better decisions, be skeptical and thorough about their assumptions, and, as we hope to convince you, make the process easier for themselves, because it requires methodical organization that is labor-saving and efficient over the complete course of a project.

[27] http://www.princeton.edu/~mjs3/open_and_reproducible_opr_2017.pdf

[28] https://dimewiki.worldbank.org/wiki/Data_Documentation

Tools like pre-registration, pre-analysis plans, and **Registered Reports**[29] can help with this process where they are available. By pre-specifying a large portion of the research design,[30] a great deal of analytical planning has already been completed, and at least some research questions are pre-committed for publication regardless of the outcome. This is meant to combat the "file-drawer problem",[31] and ensure that researchers are transparent in the additional sense that all the results obtained from registered studies are actually published. In no way should this be viewed as binding the hands of the researcher: anything outside the original plan is just as interesting and valuable as it would have been if the the plan was never published; but having pre-committed to any particular inquiry makes its results immune to a wide range of criticisms of specification searching or multiple testing.

Documenting a project in detail greatly increases transparency.

[29] https://blogs.worldbank.org/impactevaluations/registered-reports-piloting-pre-results-review-process-journal-development-economics

[30] https://www.bitss.org/2019/04/18/better-pre-analysis-plans-through-design-declaration-and-diagnosis/

[31] Simonsohn, U., Nelson, L. D., and Simmons, J. P. (2014). P-curve: a key to the file-drawer. *Journal of Experimental Psychology: General*, 143(2):534

Many disciplines have a tradition of keeping a "lab notebook", and adapting and expanding this process for the development of lab-style working groups in development is a critical step. This means explicitly noting decisions as they are made, and explaining the process behind the decision-making. Documentation on data processing and additional hypotheses tested will be expected in the supplemental materials to any publication. Careful documentation will also save the research team a lot of time during a project, as it prevents you from having the same discussion twice (or more!), since you have a record of why something was done in a particular way. There are a number of available tools that will contribute to producing documentation, but project documentation should always be an active and ongoing process, not a one-time requirement or retrospective task. New decisions are always being made as the plan begins contact with reality, and there is nothing wrong with sensible adaptation so long as it is recorded and disclosed. (Email is *not* a note-taking service, because communications are rarely well-ordered, can be easily deleted, and are not available for future team members.)

There are various software solutions for building documentation over time. The **Open Science Framework**[32] provides one such solution, with integrated file storage, version histories, and collaborative wiki pages. **GitHub**[33] provides a transparent documentation system[34], in addition to version histories and wiki pages. Such services offers multiple different ways to record the decision process leading to changes and additions, track and register discussions, and manage tasks. These are flexible tools that can be adapted to different team and project dynamics. Each project has specific requirements for data, code, and documentation management, and the exact shape of this process can be molded to the team's needs, but it should be agreed upon prior to project launch. This way, you can start building a project's documentation as soon as you start making decisions.

[32] https://osf.io/

[33] https://github.com

[34] https://dimewiki.worldbank.org/wiki/Getting_started_with_GitHub

Research credibility

The credibility of research is traditionally a function of design choices.[35] Is the research design sufficiently powered through its sampling and randomization? Were the key research outcomes pre-specified or chosen ex-post? How sensitive are the results to changes in specifications or definitions? Tools such as **pre-analysis plans**[36] can be used to assuage these concerns for experimental evaluations by fully specifying some set of analysis intended to be conducted, but they may feel like "golden handcuffs" for other types of research.[37] Regardless of whether or not a formal pre-analysis plan is utilized, all experimental and observational studies should be **pre-registered**[38]

[35] Angrist, J. D. and Pischke, J.-S. (2010). The credibility revolution in empirical economics: How better research design is taking the con out of econometrics. *Journal of Economic Perspectives*, 24(2):3–30; and Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Medicine*, 2(8):e124

[36] https://dimewiki.worldbank.org/wiki/Pre-Analysis_Plan

[37] Olken, B. A. (2015). Promises and perils of pre-analysis plans. *Journal of Economic Perspectives*, 29(3):61–80

[38] https://dimewiki.worldbank.org/wiki/Pre-Registration

simply to create a record of the fact that the study was undertaken.[39] This is increasingly required by publishers and can be done very quickly using the **AEA** database[40], the **3ie** database[41], the **eGAP** database[42], or the **OSF** registry[43] as appropriate.

Common research standards from journals, funders, and others feature both ex ante (or "regulation") and ex post (or "verification") policies.[44] Ex ante policies require that authors bear the burden of ensuring they provide some set of materials before publication and their quality meet some minimum standard. Ex post policies require that authors make certain materials available to the public, but their quality is not a direct condition for publication. Still, others have suggested "guidance" policies that would offer checklists for which practices to adopt, such as reporting on whether and how various practices were implemented.[45]

With the ongoing rise of empirical research and increased public scrutiny of scientific evidence, simply making analysis code and data available is no longer sufficient on its own to guarantee that findings will hold their credibility. Even if your methods are highly precise, your evidence is only as good as your data – and there are plenty of mistakes that can be made between establishing a design and generating final results that would compromise its conclusions. That is why transparency is key for research credibility. It allows other researchers, and research consumers, to verify the steps to a conclusion by themselves, and decide whether their standards for accepting a finding as evidence are met. Therefore we encourage you to work, gradually, towards improving the documentation and release of your research materials, and finding the tools and workflows that best match your project and team. Every investment you make in documentation and transparency up front protects your project down the line, particularly as these standards continue to tighten. Since projects tend to span over many years, the records you will need to have available for publication are only bound to increase by the time you do so.

[39] http://datacolada.org/12

[40] https://www.socialscienceregistry.org/

[41] http://ridie.3ieimpact.org/

[42] http://egap.org/content/registration/

[43] https://osf.io/registries

[44] Stodden, V., Guo, P., and Ma, Z. (2013). Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals. *PloS one*, 8(6):e67111

[45] Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., et al. (2015). Promoting an open research culture. *Science*, 348(6242):1422–1425

Ensuring privacy and security in research data

Anytime you are collecting primary data in a development research project, you are almost certainly handling data that include **personally-identifying information (PII)**[46]. PII data contains information that can, without any transformation, be used to identify individual people, households, villages, or firms that were included in **data collection**. This includes names, addresses, and geolocations, and extends to personal information such as email addresses, phone numbers, and financial information. It is important to keep in mind

[46] **Personally-identifying information:** any piece or set of information that can be used to identify an individual research subject. https://dimewiki.worldbank.org/wiki/De-identification#Personally_Identifiable_Information

these data privacy principles not only for the individual respondent but also the PII data of their household members or other caregivers who are covered under the survey. In some contexts this list may be more extensive – for example, if you are working in an environment that is either small, specific, or has extensive linkable data sources available to others, information like someone's age and gender may be sufficient to identify them even though these would not be considered PII in a larger context. There is no one-size-fits-all solution to determine what is PII, and you will have to use careful judgment in each case to decide which pieces of information fall into this category.[47]

[47] https://sdcpractice.readthedocs.io/en/latest/

In all cases where this type of information is involved, you must make sure that you adhere to several core principles. These include ethical approval, participant consent, data security, and participant privacy. If you are a US-based researcher, you will become familiar with a set of governance standards known as "The Common Rule".[48] If you interact with European institutions or persons, you will also become familiar with "GDPR",[49] a set of regulations governing **data ownership** and privacy standards.[50] In all settings, you should have a clear understanding of who owns your data (it may not be you, even if you collect or possess it), the rights of the people whose information is reflected there, and the necessary level of caution and risk involved in storing and transferring this information. Due to the increasing scrutiny on many organizations from recently advanced data rights and regulations, these considerations are critically important. Check with your organization if you have any legal questions; in general, you are responsible to avoid taking any action that knowingly or recklessly ignores these considerations.

[48] https://www.hhs.gov/ohrp/regulations-and-policy/regulations/common-rule/index.html

[49] http://blogs.lshtm.ac.uk/library/2018/01/15/gdpr-for-research-data/

[50] **Data ownership:** the set of rights governing who may accesss, alter, use, or share data, regardless of who possesses it.

Obtaining ethical approval and consent

For almost all data collection or research activities that involves PII data, you will be required to complete some form of **Institutional Review Board (IRB)** process.[51] Most commonly this consists of a formal application for approval of a specific protocol for consent, data collection, and data handling.[52] An IRB which has sole authority over your project is not always apparent, particularly if some institutions do not have their own. It is customary to obtain an approval from a university IRB where at least one PI is affiliated, and if work is being done in an international setting, approval is often also required from an appropriate institution subject to local law.

[51] **Institutional Review Board (IRB):** An institution formally responsible for ensuring that research meets ethical standards.

[52] https://dimewiki.worldbank.org/wiki/IRB_Approval

One primary consideration of IRBs is the protection of the people about whom information is being collected and whose lives may be affected by the research design. Some jurisdictions (especially those

responsible to EU law) view all personal data as being intrinsically owned by the persons who they describe. This means that those persons have the right to refuse to participate in data collection before it happens, as it is happening, or after it has already happened. It also means that they must explicitly and affirmatively consent to the collection, storage, and use of their information for any purpose. Therefore, the development of appropriate consent processes is of primary importance. Ensuring that research participants are aware that their information will be stored and may be used for various research purposes is critical. There are special additional protections in place for vulnerable populations, such as minors, prisoners, and people with disabilities, and these should be confirmed with relevant authorities if your research includes them.

Make sure you have significant advance timing with your IRB submissions. You may not begin data collection until approval is in place, and IRBs may have infrequent meeting schedules or require several rounds of review for an application to be approved. If there are any deviations from an approved plan or expected adjustments, report these as early as you can so that you can update or revise the protocol. Particularly at universities, IRBs have the power to retroactively deny the right to use data which was not collected in accordance with an approved plan. This is extremely rare, but shows the seriousness of these considerations since the institution itself may face legal penalties if its IRB is unable to enforce them. As always, as long as you work in good faith, you should not have any issues complying with these regulations.

Transmitting and storing data securely

Secure data storage and transfer are ultimately your personal responsibility.[53] First, all online and offline accounts – including personal accounts like computer logins and email – need to be protected by strong and unique passwords. There are several services that create and store these passwords for you, and some provide utilities for sharing passwords with others inside that secure environment if multiple users share accounts. However, password-protection alone is not sufficient, because if the underlying data is obtained through a leak the information itself remains usable. Raw data which contains PII *must* therefore be **encrypted**[54] during data collection, storage, and transfer. The biggest security gap is often in transmitting survey plans to and from staff in the field, since staff with technical specialization are usually in an HQ office. To protect information in transit to field staff, some key steps are: (a) to ensure that all devices that store PII data have hard drive encryption and

[53] https://dimewiki.worldbank.org/wiki/Data_Security

[54] **Encryption:** Methods which ensure that files are unreadable even if laptops are stolen, databases are hacked, or any other type of unauthorized access is obtained. https://dimewiki.worldbank.org/wiki/encryption

password-protection; (b) that no PII information is sent over e-mail, WhatsApp, etc. without encrypting the information first; and (c) all field staff receive adequate training on the privacy standards applicable to their work.

Most modern data collection software has features that, if enabled, make secure transmission straightforward.[55] Many also have features that ensure data is encrypted when stored on their servers, although this usually needs to be actively enabled and administered. Proper encryption means that, even if the information were to be intercepted or made public, the files that would be obtained would be useless to the recipient. In security language this person is often referred to as an "intruder" but it is rare that data breaches are malicious or even intentional.

[55] https://dimewiki.worldbank.org/wiki/SurveyCTO_Form_Settings

The easiest way to protect personal information is not to use it. It is often very simple to conduct planning and analytical work using a subset of the data that has anonymous identifying ID variables, and has had personal characteristics removed from the dataset altogether. We encourage this approach, because it is easy. However, when PII is absolutely necessary for work, such as geographical location, application of intervention programs, or planning or submission of survey materials, you must actively protect those materials in transmission and storage.

There are plenty of options available to keep your data safe, at different prices, from enterprise-grade solutions to free software. It may be sufficient to hold identifying information in an encrypted service, or you may need to encrypt information at the file level using a special tool. (This is in contrast to using software or services with disk-level or service-level encryption.) Data security is important not only for identifying, but also sensitive information, especially when a worst-case scenario could potentially lead to re-identifying subjects. Extremely sensitive information may be required to be held in a "cold" machine which does not have Internet access – this is most often the case with government records such as granular tax information. Each of these tools and requirements will vary in level of security and ease of use, and sticking to a standard practice will make your life much easier, so agreeing on a protocol from the start of a project is ideal. Finally, having an end-of-life plan for data is essential: you should always know how to transfer access and control to a new person if the team changes, and what the expiry of the data and the planned deletion processes are.

De-identifying and anonymizing information

Most of the field research done in development involves human subjects.[56] As a researcher, you are asking people to trust you with personal information about themselves: where they live, how rich they are, whether they have committed or been victims of crimes, their names, their national identity numbers, and all sorts of other data. PII data carries strict expectations about data storage and handling, and it is the responsibility of the research team to satisfy these expectations.[57] Your donor or employer will most likely require you to hold a certification from a source such as Protecting Human Research Participants[58] or the CITI Program.[59]

[56] https://dimewiki.worldbank.org/wiki/Human_Subjects_Approval

[57] https://dimewiki.worldbank.org/wiki/Research_Ethics

[58] https://phrptraining.com

[59] https://about.citiprogram.org/en/series/human-subjects-research-hsr/

In general, though, you shouldn't need to handle PII data very often once the data collection processes are completed. You can take simple steps to avoid risks by minimizing the handling of PII. First, only collect information that is strictly needed for the research. Second, avoid the proliferation of copies of identified data. There should only be one raw identified dataset copy and it should be somewhere where only approved people can access it. Finally, not everyone on the research team needs access to identified data. Analysis that requires PII data is rare and can be avoided by properly linking identifiers to research information such as treatment statuses and weights, then removing identifiers.

Therefore, once data is securely collected and stored, the first thing you will generally do is **de-identify** it.[60] (We will provide more detail on this in the chapter on data collection.) This will create a working de-identified copy that can safely be shared among collaborators. De-identified data should avoid, for example, you being sent back to every household to alert them that someone dropped all their personal information on a public bus and we don't know who has it. This simply means creating a copy of the data that contains no personally-identifiable information. This data should be an exact copy of the raw data, except it would be okay if it were for some reason publicly released.[61]

[60] https://dimewiki.worldbank.org/wiki/De-identification

[61] Matthews, G. J., Harel, O., et al. (2011). Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Surveys*, 5:1–29

Note, however, that it is in practice impossible to **anonymize** data. There is always some statistical chance that an individual's identity will be re-linked to the data collected about them – even if that data has had all directly identifying information removed – by using some other data that becomes identifying when analyzed together. There are a number of tools developed to help researchers de-identify data and which you should use as appropriate at that stage of data collection. These include `PII_detection`[62] from IPA, `PII-scan`[63] from JPAL, and `sdcMicro`[64] from the World Bank. The `sdcMicro` tool, in particular, has a feature that allows you to assess

[62] https://github.com/PovertyAction/PII_detection

[63] https://github.com/J-PAL/PII-Scan

[64] https://sdcpractice.readthedocs.io/en/latest/sdcMicro.html

the uniqueness of your data observations, and simple measures of the identifiability of records from that. Additional options to protect privacy in data that will become public exist, and you should expect and intend to release your datasets at some point. One option is to add noise to data, as the US Census Bureau has proposed,[65] as it makes the trade-off between data accuracy and privacy explicit. But there are no established norms for such "differential privacy" approaches: most approaches fundamentally rely on judging "how harmful" information disclosure would be. The fact remains that there is always a balance between information release (and therefore transparency) and privacy protection, and that you should engage with it actively and explicitly. The best thing you can do is make a complete record of the steps that have been taken so that the process can be reviewed, revised, and updated as necessary.

[65] Abowd, J. M. (2018). The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867. ACM

Chapter 2: Collaborating on code and data

Preparation for collaborative data work begins long before you collect any data, and involves planning both the software tools you will use yourself and the collaboration platforms and processes for your team. In order to be prepared to work on the data you receive with a group, you need to plan out the structure of your workflow in advance. This means knowing which data sets and output you need at the end of the process, how they will stay organized, what types of data you'll handle, and whether the data will require special handling due to size or privacy considerations. Identifying these details should help you map out the data needs for your project, giving you and your team a sense of how information resources should be organized. It's okay to update this map once the project is underway – the point is that everyone knows – at any given time – what the plan is.

To implement this plan, you will need to prepare collaborative tools and workflows. Changing software or protocols halfway through a project can be costly and time-consuming, so it's important to think ahead about decisions that may seem of little consequence. For example, things as simple as sharing services, folder structures, and filenames can be extremely painful to alter down the line in any project. Similarly, making sure to set up a self-documenting discussion platform and version control processes makes working together on outputs much easier from the very first discussion. This chapter will discuss some tools and processes that will help prepare you for collaboration and replication. We will provide free, open-source, and platform-agnostic tools wherever possible, and point to more detailed instructions when relevant. (Stata is the notable exception here due to its current popularity in the field.) Most have a learning and adaptation process, meaning you will become most comfortable with each tool only by using it in real-world work. Get to know them well early on, so that you do not spend a lot of time learning through trial and error.

Preparing a collaborative work environment

Being comfortable using your computer and having the tools you need in reach is key. This section provides a brief introduction to core concepts and tools that can help you handle the work you will be primarily responsible for. Some of these skills may seem elementary, but thinking about simple things from a workflow perspective can help you make marginal improvements every day you work that add up to substantial gains over the course of many projects. Together, these processes should form a collaborative workflow that will greatly accelerate your team's ability to get tasks done on every project you take on together.

Teams often develop their workflows over time, solving new challenges as they arise. This is good. But it is important to recognize that there are a number of tasks that will exist for every project, and that their corresponding workflows can be agreed on in advance. These include documentation methods, software choices, naming schema, organizing folders and outputs, collaborating on code, managing revisions to files, and reviewing each other's work. These tasks appear in almost every project, and their solutions translate well between projects. Therefore, there are large efficiency gains over time to thinking in advance about the best way to do these tasks, instead of throwing together a solution when the task arises. This chapter will outline the main points to discuss within the team, and suggest some common solutions for these tasks.

Setting up your computer

First things first: turn on your computer. Make sure you have fully updated the operating system, that it is in good working order, and that you have a **password-protected** login. All machines should have **hard disk encryption** enabled. Disk encryption is built-in on most modern operating systems; the service is currently called BitLocker on Windows or FileVault on MacOS. Disk encryption prevents your files from ever being accessed without first entering the system password. This is different from file-level encryption, which makes individual files unreadable without a specific key. (We will address that in more detail later.) As with all critical passwords, your system password should be strong, memorable, and backed up in a separate secure location.

Make sure your computer is backed up to prevent information loss. Follow the **3-2-1 rule**: maintain 3 copies of all original or irreplaceable data, on at least 2 different hardware devices you have access to, with 1 offsite storage method.[66] One example of this setup is having one copy on your primary computer, one copy on an external hard drive stored in a safe place, and one copy in the cloud. In this case, Dropbox and other automatic file sync services do not count as a cloud copy, since other users can alter or delete them unless you create a specific folder for this purpose that is not shared with anyone else.

[66] https://www.backblaze.com/blog/the-3-2-1-backup-strategy/

Ensure you know how to get the **absolute file path** for any given file. Using the absolute file path, starting from the filesystem root, means that the computer will never accidentally load the wrong file. On MacOS this will be something like `/users/username/git/project/...`, and on Windows, `C:/users/username/git/project/...`. Use forward slashes (`/`) in filepaths for folders, and whenever

possible use only A-Z (the 26 English characters), dashes (-), and underscores (_) in folder names and filenames. For emphasis: *always* use forward slashes (/) in file paths in code, just like in internet addresses. Do this even if you are using a Windows machine where both forward and backward slashes are allowed, as your code will otherwise break if anyone tries to run it on a Mac or Linux machine. Making the structure of your directories a core part of your workflow is very important, since otherwise you will not be able to reliably transfer the instructions for replicating or carrying out your analytical work.

When you are working with others, you will most likely be using some kind of **file sharing** software. The exact services you use will depend on your tasks, but in general, there are different approaches to file sharing, and the three discussed here are the most common. **File syncing** is the most familiar method, and is implemented by software like Dropbox and OneDrive. Sync forces everyone to have the same version of every file at the same time, which makes simultaneous editing difficult but other tasks easier. They also have some security concerns which we will address later. **Version control** is another method, commonly implemented by tools like Git[67] and GitHub[68]. Version control allows everyone to access different versions of files at the same time, making simultaneous editing easier but some other tasks harder. It is also only optimized for specific types of files. Finally, **server storage** is the least-common method, because there is only one version of the materials, and simultaneous access must be carefully regulated. Server storage ensures that everyone has access to exactly the same files and environment, and it also enables high-powered computing processes for large and complex data. All three file sharing methods are used for collaborative workflows, and you should review the types of data work that you will be doing, and plan which types of files will live in which types of sharing services. It is important to note that they are, in general, not interoperable: meaning you should not have version-controlled files inside a syncing service, or vice versa, without setting up complex workarounds, and you cannot shift files between them without losing historical information. Therefore, choosing the correct sharing service at the outset is essential.

[67] **Git:** a multi-user version control system for collaborating on and tracking changes to code as it is written.

[68] **GitHub:** the biggest publicly available platform for hosting Git projects.

Documenting decisions and tasks

Once your technical and sharing workspace is set up, you need to decide how you are going to communicate with your team. The first habit that many teams need to break is using instant communication for management and documentation. Email is, simply put, not a

system. It is not a system for anything. Neither is WhatsApp. These tools are developed for communicating "now" and this is what they do well. They are not structured to manage group membership or to present the same information across a group of people, or to remind you when old information becomes relevant. They are not structured to allow people to collaborate over a long time or to review old discussions. It is therefore easy to miss or lose communications from the past when they have relevance in the present. Everything with future relevance that is communicated over e-mail or any other instant medium – such as, for example, decisions about sampling – should immediately be recorded in a system that is designed to keep permanent records. We call these systems collaboration tools, and there are several that are very useful.[69]

[69] https://dimewiki.worldbank.org/wiki/Collaboration_Tools

Many collaboration tools are web-based so that everyone on your team can access them simultaneously and have live discussions about tasks and processes. Many are based on an underlying system known as "Kanban".[70] This task-oriented system allows the team to create and assign tasks, carry out discussions related to single tasks, track task progress across time, and quickly see the overall project state. These systems therefore link communication to specific tasks so that the records related to decision making on those tasks is permanently recorded and easy to find in the future when questions about that task come up. One popular and free implementation of this system is found in GitHub project boards. Other tools which currently offer similar features (but are not explicitly Kanban-based) are GitHub Issues and Dropbox Paper. Any specific list of software will quickly be outdated; we mention these two as an example of one that is technically-organized and one that is chronologial. Choosing the right tool for the right needs is essential to being satisfied with the workflow. What is important is that your team chooses its systems and stick to those choices, so that decisions, discussions, and tasks are easily reviewable long after they are completed.

[70] https://en.wikipedia.org/wiki/Kanban_board

Just like we use different file sharing tools for different types of files, we can use different collaboration tools for different types of tasks. Our team, for example, uses GitHub Issues for code-related tasks, and Dropbox Paper for more managerial and office-related tasks. GitHub creates incentives for writing down why changes were made in response to specific discussions as they are completed, creating naturally documented code. It is useful also because tasks in Issues can clearly be tied to file versions. On the other hand, Dropbox Paper provides a clean interface with task notifications, and is very intuitive for people with non-technical backgrounds. It is useful because tasks can be easily linked to other documents saved in Dropbox. Therefore, it is a better tool for managing non-code-related

tasks. Neither of these tools require much technical knowledge; they merely require an agreement and workflow design so that the people assigning the tasks are sure to set them up in the appropriate system.

Choosing software

Choosing the right working environments can make your work significantly easier. It may be difficult or costly to switch halfway through a project, so think ahead about the different software to be used. Take into account the different levels of techiness of team members, how important it is to access files offline constantly, as well as the type of data you will need to access and the security needed. Big datasets require additional infrastructure and may overburden the traditional tools used for small datasets, particularly if you are trying to sync or collaborate on them. Also consider the cost of licenses, the time to learn new tools, and the stability of the tools. There are few strictly right or wrong choices for software, but what is important is that you have a plan in advance and understand how your tools will interact with your work.

Ultimately, the goal is to ensure that you will be able to hold your code environment constant over the lifecycle of a single project. While this means you will inevitably have different projects with different code environments, each one will be better than the last, and you will avoid the extremely costly process of migrating a project into a new code enviroment while it is still ongoing. This can be set up down to the software level: you should ensure that even specific versions of software and the individual packages you use are referenced or maintained so that they can be reproduced going forward even if their most recent releases contain changes that would break your code. (For example, our command `ieboilstart` in the `ietoolkit` package provides functionality to support Stata version stability.[71])

[71] https://dimewiki.worldbank.org/wiki/ieboilstart

Next, think about how and where you write and execute code. This book focuses mainly on primary survey data, so we are going to broadly assume that you are using "small" datasets in one of the two most popular desktop-based packages: R or Stata. (If you are using another language, like Python, or working with big data projects on a server installation, many of the same principles apply but the specifics will be different.) The most visible part of working with code is a code editor, since most of your time will be spent writing and re-writing your code. This does not need to be the same program as the code runs in, and the various members of your team do not need to use the same editor. Using an external editor can be preferable since your editor will not crash if your code does, and

may offer additional features aimed at writing code well. If you are working in R, **RStudio** is the typical choice.[72] For Stata, the built-in do-file editor is the most widely adopted code editor, but **Atom**[73] and **Sublime**[74] can also be configured to run Stata code externally, while offering great code accessibility and quality features. (We recommend setting up and becoming comfortable with one of these.) For example, these editors can access an entire directory – rather than a single file – which gives you access to directory views and file management actions, such as folder management, Git integration, and simultaneous work with other types of files, without leaving the editor.

[72] https://www.rstudio.com

[73] https://atom.io

[74] https://www.sublimetext.com/

In our field of development economics, Stata is currently the most commonly used statistical software, and the built-in do-file editor the most common editor for programming Stata. We focus on Stata-specific tools and instructions in this book. Hence, we will use the terms 'script' and 'do-file' interchangeably to refer to Stata code throughout. This is only in part due to its popularity. Stata is primarily a scripting language for statistics and data, meaning that its users often come from economics and statistics backgrounds and understand Stata to be encoding a set of tasks as a record for the future. We believe that this must change somewhat: in particular, we think that practitioners of Stata must begin to think about their code and programming workflows just as methodologically as they think about their research workflows. and that people who adopt this approach will be dramatically more capable in their analytical ability. This means that they will be more productive when managing teams, and more able to focus on the challenges of experimental design and econometric analysis, rather than spending excessive time re-solving problems on the computer. To support this goal, this book also includes an introductory Stata Style Guide that we use in our work, which provides some new standards for coding so that code styles can be harmonized across teams for easier understanding and reuse of code. Stata also has relatively few resources of this type available, and the ones that we have created and shared here we hope will be an asset to all its users.

Organizing code and data

Organizing files and folders is not a trivial task. What is intuitive to one person rarely comes naturally to another, and searching for files and folders is everybody's least favorite task. As often as not, you come up with the wrong one, and then it becomes very easy to create problems that require complex resolutions later. This section will provide basic tips on managing the folder that will store your

project's data work.

We assume you will be working with code and data throughout your project. We further assume you will want all your processes to be recorded and easily findable at any point in time. Maintaining an organized file structure for data work is the best way to ensure that you, your teammates, and others are able to easily advance, edit, and replicate your work in the future. It also ensures that automated processes from code and scripting tools are able to interact well with your work, whether they are yours or those of others. File organization makes your own work easier as well as more transparent, and will make your code easier to combine with tools like version control systems that aim to cut down on the amount of repeated tasks you have to perform. It is worth thinking in advance about how to store, name, and organize the different types of files you will be working with, so that there is no confusion down the line and everyone has the same expectations.

Organizing files and folder structures

Agree with your team on a specific directory structure, and set it up at the beginning of the research project in your root folder (the one over which you can control access permissions). This will prevent future folder reorganizations that may slow down your workflow and, more importantly, ensure that your code files are always able to run on any machine. To support consistent folder organization, DIME Analytics maintains `iefolder`[75] as a part of our `ietoolkit` package. This Stata command sets up a pre-standardized folder structure for what we call the `DataWork` folder.[76] The `DataWork` folder includes folders for all the steps of a typical project. Since each project will always have its own needs, we have tried to make it as easy as possible to adapt when that is the case. The main advantage of having a universally standardized folder structure is that changing from one project to another requires less time to get acquainted with a new organization scheme. For our group, maintaining a single unified directory structure across the entire portfolio of projects means that everyone can easily move between projects without having to reorient themselves to how files and folders are organized.

[75] https://dimewiki.worldbank.org/wiki/iefolder

[76] https://dimewiki.worldbank.org/wiki/DataWork_Folder

Our suggested file structure is not for everyone. But if you do not already have a standard file structure across projects, it is intended to be an easy template to start from. This system operates by creating a `DataWork` folder at the project level, and within that folder, it provides standardized directory structures for each data source (in the primary data context, "rounds" of data collection). For each, `iefolder` creates folders for raw encrypted data, raw deidentified data, cleaned data,

final data, outputs, and documentation. In parallel, it creates folders for the code files that move the data through this progression, and for the files that manage final analytical work. The command also has some flexibility for the addition of folders for non-primary data sources, although this is less well developed. The `ietoolkit` package also includes the `iegitaddmd` command, which can place `README.md` placeholder files in your folders so that your folder structure can be shared using Git. Since these placeholder files are in **Markdown** they also provide an easy way to document the contents of every folder in the structure.

The `DataWork` folder may be created either inside an existing project-based folder structure, or it may be created separately. It's usually created by the leading RA in agreement with the PI. Increasingly, our recommendation is to create the `DataWork` folder separately from the project management materials, reserving the "project folder" for contracts, Terms of Reference, briefs and other administrative or management work. This is so the project folder can be maintained in a synced location like Dropbox, while the code folder can be maintained in a version-controlled location like GitHub. (Remember, a version-controlled folder *should not* be stored in a synced folder that is shared with other people. Those two types of collaboration tools function very differently and will almost always create undesired functionality if combined.) Nearly all code files and raw outputs (not datasets) are best managed this way. This is because code files are always **plaintext** files, and non-technical files are usually **binary** files. It's also becoming more and more common for written outputs such as reports, presentations and documentations to be written using plaintext tools such as LaTeX and dynamic documents. Keeping such plaintext files in a version-controlled folder allows you to maintain better control of their history and functionality. Because of the high degree of dependence between code files depend and file structure, you will be able to enforce better practices in a separate folder than in the project folder, which will usually be managed by a PI, FC, or field team members.

Setting up the `DataWork` folder folder in a version-controlled directory also enables you to use Git and GitHub for version control on your code files. A **version control system** is required to manage changes to any technical file. A good version control system tracks who edited each file and when, and additionally provides a protocol for ensuring that conflicting versions are avoided. This is important, for example, for your team to be able to find the version of a presentation that you delivered to a donor, or to understand why the significance level of your estimates has changed. Everyone who has ever encountered a file named something like `final_report_v5_LJK_KLE_jun15.docx`

can appreciate how useful such a system can be.

Most syncing services offer some kind of rudimentary version control; these are usually enough to manage changes to binary files (such as Word and PowerPoint documents) without needing to rely on dreaded filename-based versioning conventions. For code files, however, a more detailed version control system is usually desirable. We recommend using Git for all code and all other plaintext files (LaTeX files, .csv/.txt tables etc.). Git tracks all the changes you make to your code, and allows you to go back to previous versions without losing the information on changes made. It also makes it possible to work on multiple parallel versions of the code, so you don't risk breaking the code for other team members as you try something new. The DIME file management and organization approach is designed with this in mind.

Once the `DataWork` folder's directory structure is set up, you should adopt a file naming convention. You will generally be working with two types of files: "technical" files, which are those that are accessed by code processes, and "non-technical" files, which will not be accessed by code processes. The former takes precedent: an Excel file is a technical file even if it is a field log, because at some point it will be used by code. We will not give much emphasis to files that are not linked to code here; but you should make sure to name them in an orderly fashion that works for your team. These rules will ensure you can find files within folders and reduce the amount of time others will spend opening files to find out what is inside them. The main point to be considered is that files accessed by code have special naming requirements[77], since different software and operating systems read file names in different ways. Some of the differences between the two naming approaches are major and may be new to you, so below are a few examples. Introducing spaces between words in a file name (including the folder path) can break a file's path when it's read by code, so while a Word document may be called `2019-10-30 Sampling Procedure Description.docx`, a related do file would have a name like `sampling-endline.do`. Adding timestamps to binary files as in the example above can be useful, as it is not straightforward to track changes using version control software. However, for plaintext files tracked using Git, timestamps are an unnecessary distraction. Similarly, technical files should never include capital letters, as strings and file paths are case-sensitive in some software. Finally, one organizational practice that takes some getting used to is the fact that the best names from a coding perspective are usually the opposite of those from an English perspective. For example, for a deidentified household dataset from the baseline round, you should prefer a name like

[77] http://www2.stat.duke.edu/~rcs46/lectures_2015/01-markdown-Git/slides/naming-slides/naming-slides.pdf

`baseline-household-deidentified.dta`, rather than the opposite way around as occurs in natural language. This ensures that all `baseline` data stays together, then all `baseline-household` data, and finally provides unique information about this specific file.

Documenting and organizing code

Once you start a project's data work, the number of scripts, datasets, and outputs that you have to manage will grow very quickly. This can get out of hand just as quickly, so it's important to organize your data work and follow best practices from the beginning. Adjustments will always be needed along the way, but if the code is well-organized, they will be much easier to make. Below we discuss a few crucial steps to code organization. They all come from the principle that code is an output by itself, not just a means to an end, and should be written thinking of how easy it will be for someone to read it later.

Code documentation is one of the main factors that contribute to readability. Start by adding a code header to every file. A code header is a long **comment**[78] that details the functionality of the entire script. This should include simple things such as the purpose of the script and the name of the person who wrote it. If you are using a version control software, the last time a modification was made and the person who made it will be recorded by that software. Otherwise, you should include it in the header. Finally, use the header to track the inputs and outputs of the script. When you are trying to track down which code creates which data set, this will be very helpful. While there are other ways to document decisions related to creating code (GitHub offers a lot of different documentation options, for example), the information that is relevant to understand the code should always be written in the code file.

[78] **Comments:** Code components that have no function, but describe in plain language what the code is supposed to do.

In the script, alongside the code, are two types of comments that should be included. The first type of comment describes what is being done. This might be easy to understand from the code itself if you know the language well enough and the code is clear, but often it is still a great deal of work to reverse-engineer the code's intent. Writing the task in plain English (or whichever language you communicate with your team on) will make it easier for everyone to read and understand the code's purpose – and also for you to think about your code as you write it. The second type of comment explains why the code is performing a task in a particular way. As you are writing code, you are making a series of decisions that (hopefully) make perfect sense to you at the time. These are often highly specialized and may exploit a functionality that is not obvious or has not been seen by others before. Even you will probably not

remember the exact choices that were made in a couple of weeks. Therefore, you must document your precise processes in your code.

Code organization means keeping each piece of code in an easily findable location. Breaking your code into independently readable "chunks" is one good practice on code organization. You should write each functional element as a chunk that can run completely on its own, to ensure that each component does not depend on a complex program state created by other code chunks that are not obvious from the immediate context. One way to do this is to create sections where a specific task is completed. So, for example, if you want to find the line in your code where a variable was created, you can go straight to `PART 2: Create new variables`, instead of reading line by line through the entire code. RStudio, for example, makes it very easy to create sections, and it compiles them into an interactive script index for you. In Stata, you can use comments to create section headers, though they're just there to make the reading easier and don't have functionality. You should also add an index in the code header by copying and pasting section titles. You can then add and navigate through them using the `find` command. Since Stata code is harder to navigate, as you will need to scroll through the document, it's particularly important to avoid writing very long scripts. Therefore, in Stata at least, you should also consider breaking code tasks down into separate do-files, since there is no limit on how many you can have, how detailed their names can be, and no advantage to writing longer files. One reasonable rule of thumb is to not write do-files that have more than 200 lines. This is an arbitrary limit, just like the standard restriction of each line to 80 characters: it seems to be "enough but not too much" for most purposes.

To bring all these smaller code files together, you must maintain a master script. A master script is the map of all your project's data work which serves as a table of contents for the instructions that you code. Anyone should be able to follow and reproduce all your work from raw data to all outputs by simply running this single script. By follow, we mean someone external to the project who has the master script and all the input data can (i) run all the code and recreate all outputs, (ii) have a general understanding of what is being done at every step, and (iii) see how codes and outputs are related. The master script is also where all the settings are established, such as versions, folder paths, functions, and constants used throughout the project.

`iefolder` creates these as master do-files.[79] Master scripts are a key element of code organization and collaboration, and we will discuss some important features soon. The master script should mimic the structure of the `DataWork` folder. This is done through

[79] https://dimewiki.worldbank.org/wiki/Master_Do-files

the creation of globals (in Stata) or string scalars (in R). These coding shortcuts can refer to subfolders, so that those folders can be referenced without repeatedly writing out their absolute file paths. Because the `DataWork` folder is shared by the whole team, its structure is the same in each team member's computer. The only difference between machines should be the path to the project root folder, i.e. the highest-level shared folder, which in the context of `iefolder` is the `DataWork` folder. This is reflected in the master script in such a way that the only change necessary to run the entire code from a new computer is to change the path to the project folder to reflect the filesystem and username. The code in `stata-master-dofile.do` shows how folder structure is reflected in a master do-file.

In order to maintain these practices and ensure they are functioning well, you should agree with your team on a plan to review code as it is written. Reading other people's code is the best way to improve your coding skills. And having another set of eyes on your code will make you more comfortable with the results you find. It's normal (and common) to make mistakes as you write your code. Reading it again to organize and comment it as you prepare it to be reviewed will help you identify them. Try to have a code review scheduled frequently, every time you finish writing a piece of code, or complete a small task. If you wait for a long time to have your code reviewed, and it gets too complex, preparation and code review will require more time and work, and that is usually the reason why this step is skipped. One other important advantage of code review if that making sure that the code is running properly on other machines, and that other people can read and understand the code easily, is the easiest way to be prepared in advance for a smooth project handover or for release of the code to the general public.

Output management

The final task that needs to be discussed with your team is the best way to manage output files. A great number of outputs will be created during the course of a project, and these will include both raw outputs such as tables and graphs and final products such as presentations, papers and reports. When the first outputs are being created, agree on where to store them, what softwares and formats to use, and how to keep track of them.

Decisions about storage of outputs are made easier by technical constraints. As discussed above, version control systems like Git are a great way to manage plaintext files, and sync softwares such as Dropbox are better for binary files. Outputs will similarly come

in these two formats, depending on your software. Binary outputs like Excel files, PDFs, PowerPoints, or Word documents can be kept in a synced folder. Raw outputs in plaintext formats like `.tex` and `.eps` can be created from most analytical software and managed with Git. Tracking plaintext outputs with Git makes it easier to identify changes that affect results. If you are re-running all of your code from the master script, the outputs will be overwritten, and any changes in coefficients and number of observations, for example, will be automatically flagged for you or a reviewer to check.

No matter what choices you make, you will need to make updates to your outputs quite frequently. And anyone who has tried to recreate a graph after a few months probably knows that it can be hard to remember where you saved the code that created it. Here, naming conventions and code organization play a key role in not re-writing scripts again and again. It is common for teams to maintain one analyisis file or folder with draft code or "exploratory analysis", which are pieces of code that are stored only to be found again in the future, but not cleaned up to be included in any final outputs yet. Once you are happy with a result or output, it should be named and moved to a dedicated location. It's typically desirable to have the names of outputs and scripts linked, so, for example, `factor-analysis.do` creates `factor-analysis-f1.eps` and so on. Document output creation in the Master script that runs these files, so that before the line that runs a particular analysis script there are a few lines of comments listing data sets and functions that are necessary for it to run, as well as all outputs created by that script.

Compiling the raw outputs from your statistical software into useful formats is the final step in producing research outputs for public consumption. Though formatted text software such as Word and PowerPoint are still prevalent, researchers are increasingly choosing to prepare final outputs like documents and presentations using LaTeX.[80] LaTeX is a document preparation system that can create both text documents and presentations. The main advantage is that LaTeX uses plaintext for all formatting, and it is necessary to learn its specific markup convention to use it. The main advantage of using LaTeX is that you can write dynamic documents, that import inputs every time they are compiled. This means you can skip the copying and pasting whenever an output is updated. Because it's written in plaintext, it's also easier to control and document changes using Git. Creating documents in LaTeX using an integrated writing environment such as TeXstudio, TeXmaker or LyX is great for outputs that focus mainly on text, but include small chunks of code and static code outputs. This book, for example, was written in LaTeX and managed on GitHub[81].

[80] https://www.latex-project.org and https://github.com/worldbank/DIME-LaTeX-Templates.

[81] https://github.com/worldbank/d4di

Another option is to use the statistical software's dynamic document engines. This means you can write both text (in Markdown) and code in the script, and the result will usually be a PDF or `html` file including code, text, and outputs. Dynamic document tools are better for including large chunks of code and dynamically created graphs and tables, but formatting these can be much trickier and less full-featured than other editors. So dynamic documents can be great for creating appendices or quick documents with results as you work on them, but are not usually considered for final papers and reports. RMarkdown[82] is the most widely adopted solution in R. There are also different options for Markdown in Stata, such as `markstat`,[83] Stata 15 dynamic documents,[84] `webdoc`,[85] and `texdoc`.[86]

Whichever options you choose, agree with your team on what tools will be used for what outputs, and where they will be stored before you start creating them. Take into account ease of use for different team members, but keep in mind that learning how to use a new tool may require some time investment upfront that will be paid off as your project advances.

[82] https://rmarkdown.rstudio.com/

[83] https://data.princeton.edu/stata/markdown

[84] https://www.stata.com/new-in-stata/markdown/

[85] http://repec.sowi.unibe.ch/stata/webdoc/index.html

[86] http://repec.sowi.unibe.ch/stata/texdoc/index.html

stata-master-dofile.do

```
/*****************************************************************************
*                          TEMPLATE MASTER DO-FILE                           *
*****************************************************************************
*                                                                            *
*   PURPOSE:     Reproduce all data work, map inputs and outputs,            *
*                facilitate collaboration                                    *
*                                                                            *
*   OUTLINE:     PART 1:  Set standard settings and install packages         *
*                PART 2:  Prepare folder paths and define programs           *
*                PART 3:  Run do files                                       *
*                                                                            *
*****************************************************************************
            PART 1:  Set standard settings and install packages
*****************************************************************************/

    if (0) {
        ssc install ietoolkit,  replace
    }

    ieboilstart, v(15.1)
    `r(version)'

/*****************************************************************************
             PART 2:  Prepare folder paths and define programs
*****************************************************************************/

    * Research Assistant folder paths
    if "`c(username)'" == "ResearchAssistant" {
        global github      "C:/Users/RA/Documents/GitHub/d4di/DataWork"
        global dropbox     "C:/Users/RA/Dropbox/d4di/DataWork"
        global encrypted   "A:/DataWork/EncryptedData"  // Always mount to A disk!
    }


   * Baseline folder globals
   global bl_encrypt       "${encrypted}/Round Baseline Encrypted"
   global bl_dt            "${dropbox}/Baseline/DataSets"
   global bl_doc           "${dropbox}/Baseline/Documentation"
   global bl_do            "${github}/Baseline/Dofiles"
   global bl_out           "${github}/Baseline/Output"

/*****************************************************************************
                          PART 3: Run do files
*****************************************************************************/

/*---------------------------------------------------------------------------
  PART 3.1:  De-identify baseline data
---------------------------------------------------------------------------
 REQUIRES:   ${bl_encrypt}/Raw Identified Data/D4DI_baseline_raw_identified.dta
 CREATES:    ${bl_dt}/Raw Deidentified/D4DI_baseline_raw_deidentified.dta
 IDS VAR:    hhid
--------------------------------------------------------------------------- */
    do "${bl_do}/Cleaning/deidentify.do"

/*---------------------------------------------------------------------------
  PART 3.2:  Clean baseline data
---------------------------------------------------------------------------
  REQUIRES:   ${bl_dt}/Raw Deidentified/D4DI_baseline_raw_deidentified.dta
  CREATES:    ${bl_dt}/Final/D4DI_baseline_clean.dta
              ${bl_doc}/Codebook baseline.xlsx
  IDS VAR:    hhid
--------------------------------------------------------------------------- */
    do "${bl_do}/Cleaning/cleaning.do"

/*---------------------------------------------------------------------------
 PART 3.3:  Construct income indicators
---------------------------------------------------------------------------
  REQUIRES:   ${bl_dt}/Final/D4DI_baseline_clean.dta
  CREATES:    ${bl_out}/Raw/D4DI_baseline_income_distribution.png
              ${bl_dt}/Intermediate/D4DI_baseline_constructed_income.dta
  IDS VAR:    hhid
--------------------------------------------------------------------------- */
    do "${bl_do}/Construct/construct_income.do"
```

Chapter 3: Evaluating impact through research design

Research design is the process of defining the methods and data that will be used to answer a specific research question. You don't need to be an expert in this, and there are lots of good resources out there that focus on designing interventions and evaluations as well as on econometric approaches. Therefore, without going into technical detail, this section will present a brief overview of the most common methods that are used in development research, particularly those that are widespread in program evaluation. These "causal inference" methods will turn up in nearly every project, so you will need to have a broad knowledge of how the methods in your project are used in order to manage data and code appropriately. The intent of this chapter is for you to obtain an understanding of the way in which each method constructs treatment and control groups, the data structures needed to estimate the corresponding effects, and some available code tools designed for each method (the list, of course, is not exhaustive).

Thinking through your design before starting data work is important for several reasons. If you do not know how to calculate the correct estimator for your study, you will not be able to assess the statistical power of your research design. You will also be unable to make decisions in the field when you inevitably have to allocate scarce resources between tasks like maximizing sample size and ensuring follow-up with specific individuals. You will save a lot of time by understanding the way your data needs to be organized in order to be able to calculate meaningful results. Just as importantly, familiarity with each of these approaches will allow you to keep your eyes open for research opportunities: many of the most interesting projects occur because people in the field recognize the opportunity to implement one of these methods in response to an unexpected event. Intuitive knowledge of your project's chosen approach will make you much more effective at the analytical part of your work.

Causality, inference, and identification

When we are discussing the types of inputs – "treatments" – commonly referred to as "programs" or "interventions", we are typically attempting to obtain estimates of program-specific **treatment effects** These are the changes in outcomes attributable to the treatment.[87] The primary goal of research design is to establish **causal identification** for an effect. Causal identification means establishing that a change in an input directly altered an outcome. When a study is well-identified, then we can say with confidence that our estimate of the treatment effect would, with an infinite amount of data, give us a precise estimate of that treatment effect. Under this condition, we can proceed to draw evidence from the limited samples we have

[87] Abadie, A. and Cattaneo, M. D. (2018). Econometric methods for program evaluation. *Annual Review of Economics*, 10:465–503

access to, using statistical techniques to express the uncertainty of not having infinite data. Without identification, we cannot say that the estimate would be accurate, even with unlimited data, and therefore cannot attribute it to the treatment in the small samples that we typically have access to. Conversely, more data is not a substitute for a well-identified experimental design. Therefore it is important to understand how exactly your study identifies its estimate of treatment effects, so you can calculate and interpret those estimates appropriately. All the study designs we discuss here use the potential outcomes framework[88] to compare a group that received some treatment to another, counterfactual group. Each of these approaches can be used in two types of designs: **experimental** designs, in which the research team is directly responsible for creating the variation in treatment, and **quasi-experimental** designs, in which the team identifies a "natural" source of variation and uses it for identification. Neither type is implicitly better or worse, and both types are capable of achieving effect identification under different contexts.

[88] Athey, S. and Imbens, G. W. (2017b). The state of applied econometrics: Causality and policy evaluation. *Journal of Economic Perspectives*, 31(2):3–32

Estimating treatment effects using control groups

The key assumption behind estimating treatment effects is that every person, facility, or village (or whatever the unit of intervention is) has two possible states: their outcomes if they do not receive some treatment and their outcomes if they do receive that treatment. Each unit's treatment effect is the individual difference between these two states, and the **average treatment effect (ATE)** is the average of all individual differences across the potentially treated population. This is the parameter that most research designs attempt to estimate. Their goal is to establish a **counterfactual**[89] for the treatment group with which outcomes can be directly compared. There are several resources that provide more or less mathematically intensive approaches to understanding how various methods do this. *Impact Evaluation in Practice* is a strong general guide to these methods.[90] *Causal Inference* and *Causal Inference: The Mixtape* provides more detailed mathematical approaches to the tools.[91] *Mostly Harmless Econometrics* and *Mastering Metrics* are excellent resources on the statistical principles behind all econometric approaches.[92]

Intuitively, the problem is as follows: we can never observe the same unit in both their treated and untreated states simultaneously, so measuring and averaging these effects directly is impossible.[93] Instead, we typically make inferences from samples. **Causal inference** methods are those in which we are able to estimate the average treatment effect without observing individual-level effects, but through some comparison of averages with a **control** group. Every

[89] **Counterfactual:** A statistical description of what would have happened to specific individuals in an alternative scenario, for example, a different treatment assignment outcome.

[90] https://www.worldbank.org/en/programs/sief-trust-fund/publication/impact-evaluation-in-practice

[91] https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/ http://scunning.com/cunningham_mixtape.pdf

[92] https://www.researchgate.net/publication/51992844_Mostly_Harmless_Econometrics_An_Empiricist's_Companion http://assets.press.princeton.edu/chapters/s10363.pdf

[93] http://www.stat.columbia.edu/~cook/qr33.pdf

research design is based on a way of comparing another set of observations – the "control" observations – against the treatment group. They all work to establish that the control observations would have been identical *on average* to the treated group in the absence of the treatment. Then, the mathematical properties of averages imply that the calculated difference in averages is equivalent to the average difference: exactly the parameter we are seeking to estimate. Therefore, almost all designs can be accurately described as a series of between-group comparisons.[94]

[94] http://nickchk.com/econ305.html

Most of the methods that you will encounter rely on some variant of this strategy, which is designed to maximize their ability to estimate the effect of an average unit being offered the treatment you want to evaluate. The focus on identification of the treatment effect, however, means there are several essential features of causal identification methods that are not common in other types of statistical and data science work. First, the econometric models and estimating equations used do not attempt to create a predictive or comprehensive model of how the outcome of interest is generated. Typically, causal inference designs are not interested in predictive accuracy, and the estimates and predictions that they produce will not be as good at predicting outcomes or fitting the data as other models. Additionally, when control variables or other variables are used in estimation, there is no guarantee that the resulting parameters are marginal effects. They can only be interpreted as correlative averages, unless there are additional sources of identification. The models you will construct and estimate are intended to do exactly one thing: to express the intention of your project's research design, and to accurately estimate the effect of the treatment it is evaluating. In other words, these models tell the story of the research design in a way that clarifies the exact comparison being made between control and treatment.

Experimental and quasi-experimental research designs

Experimental research designs explicitly allow the research team to change the condition of the populations being studied,[95] often in the form of government programs, NGO projects, new regulations, information campaigns, and many more types of interventions.[96] The classic experimental causal inference method is the **randomized control trial (RCT)**.[97] In randomized control trials, the control group is randomized – that is, from an eligible population, a random subset of units are not given access to the treatment, so that they may serve as a counterfactual for those who are. A randomized control group, intuitively, is meant to represent how things would

[95] https://dimewiki.worldbank.org/wiki/Experimental_Methods

[96] Banerjee, A. V. and Duflo, E. (2009). The experimental approach to development economics. *Annual Review of Economics*, 1(1):151–178

[97] https://dimewiki.worldbank.org/wiki/Randomized_Control_Trials

have turned out for the treated group if they had not been treated, and it is particularly effective at doing so as evidenced by its broad credibility in fields ranging from clinical medicine to development. Therefore RCTs are very popular tools for determining the causal impact of specific programs or policy interventions.[98] However, there are many other types of interventions that are impractical or unethical to effectively approach using an experimental strategy, and therefore there are limitations to accessing "big questions" through RCT approaches.[99]

[98] https://www.nobelprize.org/prizes/economic-sciences/2019/ceremony-speech/

[99] https://www.nber.org/papers/w14690.pdf

Randomized designs all share several major statistical concerns. The first is the fact that it is always possible to select a control group, by chance, which is not in fact very similar to the treatment group. This feature is called randomization noise, and all RCTs share the need to assess how randomization noise may impact the estimates that are obtained. (More detail on this later.) Second, takeup and implementation fidelity are extremely important, since programs will by definition have no effect if the population intended to be treated does not accept or does not receive the treatment. Loss of statistical power occurs quickly and is highly nonlinear: 70% takeup or efficacy doubles the required sample, and 50% quadruples it.[100] Such effects are also very hard to correct ex post, since they require strong assumptions about the randomness or non-randomness of takeup. Therefore a large amount of field time and descriptive work must be dedicated to understanding how these effects played out in a given study, and often overshadow the effort put into the econometric design itself.

[100] https://blogs.worldbank.org/impactevaluations/power-calculations-101-dealing-with-incomplete-take-up

Quasi-experimental research designs,[101] by contrast, are causal inference methods based on events not controlled by the research team. Instead, they rely on "experiments of nature", in which natural variation can be argued to approximate the type of exogenous variation in treatment availability that a researcher would attempt to create with an experiment.[102] Unlike carefully planned experimental designs, quasi-experimental designs typically require the extra luck of having access to data collected at the right times and places to exploit events that occurred in the past, or having the ability to collect data in a time and place where an event that produces causal identification occurred. Therefore, these methods often use either secondary data, or they use primary data in a cross-sectional retrospective method.

[101] https://dimewiki.worldbank.org/wiki/Quasi-Experimental_Methods

[102] DiNardo, J. (2016). Natural experiments and quasi-natural experiments. *The New Palgrave Dictionary of Economics*, pages 1–12

Quasi-experimental designs therefore can access a much broader range of questions, and with much less effort in terms of executing an intervention. However, they require in-depth understanding of the precise events the researcher wishes to address in order to know what data to use and how to model the underlying natural

experiment. Additionally, because the population exposed to such events is limited by the scale of the event, quasi-experimental designs are often power-constrained. Since the research team cannot change the population of the study or the treatment assignment, power is typically maximized by ensuring that sampling for data collection is carefully designed to match the study objectives and that attrition from the sampled groups is minimized.

Obtaining treatment effects from specific research designs

Cross-sectional designs

A cross-sectional research design is any type of study that collects data in only one time period and directly compares treatment and control groups. This type of data is easy to collect and handle because you do not need track individual across time or across data sets. If this point in time is after a treatment has been fully delivered, then the outcome values at that point in time already reflect the effect of the treatment. If the study is an RCT, the control group is randomly constructed from the population that is eligible to receive each treatment. If it is a non-randomized observational study, we present other evidence that a similar equivalence holds. Therefore, by construction, each unit's receipt of the treatment is unrelated to any of its other characteristics and the ordinary least squares (OLS) regression of outcome on treatment, without any control variables, is an unbiased estimate of the average treatment effect.

For cross-sectional RCTs, what needs to be carefully maintained in data is the treatment randomization process itself, as well as detailed field data about differences in data quality and loss to follow-up across groups.[103] Only these details are needed to construct the appropriate estimator: clustering of the standard errors is required at the level at which the treatment is assigned to observations, and variables which were used to stratify the treatment must be included as controls (in the form of strata fixed effects).[104] **Randomization inference** can be used to estimate the underlying variability in the randomization process (more on this in the next chapter). **Balance checks**[105] are often reported as evidence of an effective randomization, and are particularly important when the design is quasi-experimental (since then the randomization process cannot be simulated explicitly). However, controls for balance variables are usually unnecessary in RCTs, because it is certain that the true data-generating process has no correlation between the treatment and the balance factors.[106]

Analysis is typically straightforward *once you have a strong understanding of the randomization*. A typical analysis will include a description

[103] Athey, S. and Imbens, G. W. (2017a). The econometrics of randomized experiments. *Handbook of Economic Field Experiments*, 1:73–140

[104] https://blogs.worldbank.org/impactevaluations/impactevaluations/how-randomize-using-many-baseline-variables-guest-post-thomas-barrios

[105] **Balance checks:** Statistical tests of the similarity of treatment and control groups.

[106] https://blogs.worldbank.org/impactevaluations/should-we-require-balance-t-tests-baseline-observables-randomized-experiments

of the sampling and randomization results, with analyses such as summary statistics for the eligible population, and balance checks for randomization and sample selection. The main results will usually be a primary regression specification (with multiple hypotheses appropriately adjusted for), and additional specifications with adjustments for non-response, balance, and other potential contamination. Robustness checks might include randomization-inference analysis or other placebo regression approaches. There are a number of user-written code tools that are also available to help with the complete process of data analysis,[107] including to analyze balance[108] and to visualize treatment effects.[109] Extensive tools and methods for analyzing selective non-response are available.[110]

[107] https://toolkit.povertyactionlab.org/resource/coding-resources-randomized-evaluations

[108] https://dimewiki.worldbank.org/wiki/iebaltab

[109] https://dimewiki.worldbank.org/wiki/iegraph

[110] https://blogs.worldbank.org/impactevaluations/dealing-attrition-field-experiments

Difference-in-differences

Where cross-sectional designs draw their estimates of treatment effects from differences in outcome levels in a single measurement, **differences-in-differences**[111] designs (abbreviated as DD, DiD, diff-in-diff, and other variants) estimate treatment effects from *changes* in outcomes between two or more rounds of measurement. In these designs, three control groups are used – the baseline level of treatment units, the baseline level of non-treatment units, and the endline level of non-treatment units.[112] The estimated treatment effect is the excess growth of units that receive the treatment, in the period they receive it: calculating that value is equivalent to taking the difference in means at endline and subtracting the difference in means at baseline (hence the singular "difference-in-differences").[113] The regression model includes a control variable for treatment assignment, and a control variable for time period, but the treatment effect estimate corresponds to an interaction variable for treatment and time: it indicates the group of observations for which the treatment is active. This model critically depends on the assumption that, in the absense of the treatment, the outcome of the two groups would have changed at the same rate over time, typically referred to as the **parallel trends** assumption.[114] Experimental approaches satisfy this requirement in expectation, but a given randomization should still be checked for pre-trends as an extension of balance checking.[115]

[111] https://dimewiki.worldbank.org/wiki/Difference-in-Differences

[112] https://www.princeton.edu/~otorres/DID101.pdf

[113] McKenzie, D. (2012). Beyond baseline and follow-up: The case for more T in experiments. *Journal of Development Economics*, 99(2):210–221

[114] https://blogs.worldbank.org/impactevaluations/often-unspoken-assumptions-behind-difference-difference-estimator-practice

[115] https://blogs.worldbank.org/impactevaluations/revisiting-difference-differences-parallel-trends-assumption-part-i-pre-trend

There are two main types of data structures for differences-in-differences: **repeated cross-sections** and **panel data**. In repeated cross-sections, each successive round of data collection contains a random sample of observations from the treated and untreated groups; as in cross-sectional designs, both the randomization and sampling processes are critically important to maintain alongside

the data. In panel data structures, we attempt to observe the exact same units in different points in time, so that we see the same individuals both before and after they have received treatment (or not).[116] This allows each unit's baseline outcome (the outcome before the intervention) to be used as an additional control for its endline outcome (the last outcome observation in the data), a **fixed effects** design often referred to as an ANCOVA model, which can provide large increases in power and robustness.[117] When tracking individuals over time for this purpose, maintaining sampling and tracking records is especially important, because attrition and loss to follow-up will remove that unit's information from all points in time, not just the one they are unobserved in. Panel-style experiments therefore require a lot more effort in field work for studies that use primary data.[118] Since baseline and endline may be far apart in time, it is important to create careful records during the first round so that follow-ups can be conducted with the same subjects, and attrition across rounds can be properly taken into account.[119]

As with cross-sectional designs, difference-in-differences designs are widespread. Therefore there exist a large number of standardized tools for analysis. Our `ietoolkit` Stata package includes the `ieddtab` command which produces standardized tables for reporting results.[120] For more complicated versions of the model (and they can get quite complicated quite quickly), you can use an online dashboard to simulate counterfactual results.[121] As in cross-sectional designs, these main specifications will always be accompanied by balance checks (using baseline values), as well as randomization, selection, and attrition analysis. In trials of this type, reporting experimental design and execution using the CONSORT style is common in many disciplines and will help you to track your data over time.[122]

[116] https://blogs.worldbank.org/impactevaluations/what-are-we-estimating-when-we-estimate-difference-differences

[117] https://blogs.worldbank.org/impactevaluations/another-reason-prefer-ancova-dealing-changes-measurement-between-baseline-and-follow

[118] https://www.princeton.edu/~otorres/Panel101.pdf

[119] http://blogs.worldbank.org/impactevaluations/dealing-attrition-field-experiments

[120] https://dimewiki.worldbank.org/wiki/ieddtab

[121] https://blogs.worldbank.org/impactevaluations/econometrics-sandbox-event-study-designs-co

[122] Schulz, K. F., Altman, D. G., and Moher, D. (2010). Consort 2010 statement: updated guidelines for reporting parallel group randomised trials. *BMC medicine*, 8(1):18

Regression discontinuity

Regression discontinuity (RD) designs exploit sharp breaks or limits in policy designs to separate a group of potentially eligible recipients into comparable gorups of individuals who do and do not receive a treatment.[123] These designs differ from cross-sectional and diff-in-diff designs in that the group eligible to receive treatment is not defined directly, but instead created during the treatment implementation. In an RD design, there is typically some program or event that has limited availability due to practical considerations or policy choices and is therefore made available only to individuals who meet a certain threshold requirement. The intuition of this design is that there is an underlying **running variable** that serves as the sole determinant of access to the program, and a strict cutoff determines

[123] https://dimewiki.worldbank.org/wiki/Regression_Discontinuity

the value of this variable at which eligibility stops.[124] Common examples are test score thresholds and income thresholds.[125] The intuition is that individuals who are just above the threshold will be very nearly indistinguishable from those who are just under it, and their post-treatment outcomes are therefore directly comparable.[126] The key assumption here is that the running variable cannot be directly manipulated by the potential recipients. If the running variable is time (what is commonly called an "event study"), there are special considerations.[127] Similarly, spatial discontinuity designs are handled a bit differently due to their multidimensionality.[128]

Regression discontinuity designs are, once implemented, very similar in analysis to cross-sectional or difference-in-differences designs. Depending on the data that is available, the analytical approach will center on the comparison of individuals who are narrowly on the inclusion side of the discontinuity, compared against those who are narrowly on the exclusion side.[129] The regression model will be identical to the matching research designs, i.e., contingent on whether data has one or more time periods and whether the same units are known to be observed repeatedly. The treatment effect will be identified, however, by the addition of a control for the running variable – meaning that the treatment effect estimate will only be applicable for observations in a small window around the cutoff: in the lingo, the treatment effects estimated will be "local" rather than "average". In the RD model, the functional form of the running variable control and the size of that window, often referred to as the choice of **bandwidth** for the design, are the critical parameters for the result.[130] Therefore, RD analysis often includes extensive robustness checking using a variety of both functional forms and bandwidths, as well as placebo testing for non-realized locations of the cutoff.[131]

In the analytical stage, regression discontinuity designs often include a large component of visual evidence presentation.[132] These presentations help to suggest both the functional form of the underlying relationship and the type of change observed at the discontinuity, and help to avoid pitfalls in modeling that are difficult to detect with hypothesis tests.[133] Because these designs are so flexible compared to others, there is an extensive set of commands that help assess the efficacy and results from these designs under various assumptions.[134] These packages support the testing and reporting of robust plotting and estimation procedures, tests for manipulation of the running variable, and tests for power, sample size, and randomization inference approaches that will complement the main regression approach used for point estimates.

[124] Imbens, G. W. and Lemieux, T. (2008). Regression discontinuity designs: A guide to practice. *Journal of Econometrics*, 142(2):615–635

[125] http://blogs.worldbank.org/impactevaluations/regression-discontinuity-porn

[126] Lee, D. S. and Lemieux, T. (2010). Regression discontinuity designs in economics. *Journal of Economic Literature*, 48(2):281–355

[127] Hausman, C. and Rapson, D. S. (2018). Regression discontinuity in time: Considerations for empirical applications. *Annual Review of Resource Economics*, 10:533–552

[128] https://blogs.worldbank.org/impactevaluations/spatial-jumps

[129] https://cattaneo.princeton.edu/books/Cattaneo-Idrobo-Titiunik_2019_CUP-Vol1.pdf

[130] Calonico, S., Cattaneo, M. D., Farrell, M. H., and Titiunik, R. (2019). Regression discontinuity designs using covariates. *Review of Economics and Statistics*, 101(3):442–451

[131] https://www.mdrc.org/sites/default/files/RDD%20Guide_Full%20rev%202016_0.pdf

[132] http://faculty.smu.edu/kyler/courses/7312/presentations/baumer/Baumer_RD.pdf

[133] http://econ.lse.ac.uk/staff/spischke/ec533/RD.pdf

[134] https://sites.google.com/site/rdpackages/

Instrumental variables

Instrumental variables (IV) designs, unlike the previous approaches, begin by assuming that the treatment delivered in the study in question is linked to the outcome, so its effect is not directly identifiable. Instead, similar to regression discontinuity designs, IV attempts to focus on a subset of the variation in treatment uptake and assesses that limited window of variation that can be argued to be unrelated to other factors.[135] To do so, the IV approach selects an **instrument** for the treatment status – an otherwise-unrelated predictor of exposure to treatment that affects the uptake status of an individual.[136] Whereas regression discontinuity designs are "sharp" – treatment status is completely determined by which side of a cutoff an individual is on – IV designs are "fuzzy", meaning that they do not completely determine the treatment status but instead influence the *probability* of treatment.

[135] Angrist, J. D. and Krueger, A. B. (2001). Instrumental variables and the search for identification: From supply and demand to natural experiments. *Journal of Economic Perspectives*, 15(4):69–85

[136] https://dimewiki.worldbank.org/wiki/instrumental_variables

As in regression discontinuity designs, the fundamental form of the regression is similar to either cross-sectional or difference-in-differences designs. However, instead of controlling for the instrument directly, the IV approach typically uses the **two-stage-least-squares (2SLS)** estimator.[137] This estimator forms a prediction of the probability that the unit receives treatment based on a regression against the instrumental variable. That prediction will, by assumption, be the portion of the actual treatment that is due to the instrument and not any other source, and since the instrument is unrelated to all other factors, this portion of the treatment can be used to assess its effects. Unfortunately, these estimators are known to have very high variances relative other methods, particularly when the relationship between the instrument and the treatment is small.[138] IV designs furthermore rely on strong but untestable assumptions about the relationship between the instrument and the outcome.[139] Therefore IV designs face intense scrutiny on the strength and exogeneity of the instrument, and tests for sensitivity to alternative specifications and samples are usually required with an instrumental variables analysis. However, the method has special experimental cases that are significantly easier to assess: for example, a randomized treatment *assignment* can be used as an instrument for the eventual uptake of the treatment itself, especially in cases where uptake is expected to be low, or in circumstances where the treatment is available to those who are not specifically assigned to it ("encouragement designs").

[137] http://www.nuff.ox.ac.uk/teaching/economics/bond/IV%20Estimation%20Using%20Stata.pdf

[138] Young, A. (2017). Consistency without inference: Instrumental variables in practical application. *Unpublished manuscript, London: London School of Economics and Political Science. Retrieved from: http://personal.lse.ac.uk/YoungA*

[139] Bound, J., Jaeger, D. A., and Baker, R. M. (1995). Problems with instrumental variables estimation when the correlation between the instruments and the endogenous explanatory variable is weak. *Journal of the American Statistical Association*, 90(430):443–450

In practice, there are a variety of packages that can be used to analyse data and report results from instrumental variables designs. While the built-in Stata command `ivregress` will often be used to

create the final results, the built-in packages are not sufficient on their own. The **first stage** of the design should be extensively tested, to demonstrate the strength of the relationship between the instrument and the treatment variable being instrumented.[140] This can be done using the `weakiv` and `weakivtest` commands.[141] Additionally, tests should be run that identify and exclude individual observations or clusters that have extreme effects on the estimator, using customized bootstrap or leave-one-out approaches.[142] Finally, bounds can be constructed allowing for imperfections in the exogeneity of the instrument using loosened assumptions, particularly when the underlying instrument is not directly randomized.[143]

[140] Stock, J. and Yogo, M. (2005). *Testing for Weak Instruments in Linear IV Regression*, pages 80–108. Cambridge University Press, New York

[141] https://www.carolinpflueger.com/WangPfluegerWeakivtest_20141202.pdf

[142] Young, A. (2017). Consistency without inference: Instrumental variables in practical application. *Unpublished manuscript, London: London School of Economics and Political Science. Retrieved from: http://personal.lse.ac.uk/YoungA*

[143] http://www.damianclarke.net/research/papers/practicalIV-CM.pdf

Matching

Matching methods use observable characteristics of individuals to directly construct treatment and control groups to be as similar as possible to each other, either before a randomization process or after the collection of non-randomized data.[144] Matching observations may be one-to-one or many-to-many; in any case, the result of a matching process is similar in concept to the use of randomization strata in simple randomized control trials. In this way, the method can be conceptualized as averaging across the results of a large number of "micro-experiments" in which the randomized units are verifiably similar aside from the treatment.

[144] https://dimewiki.worldbank.org/wiki/Matching

When matching is performed before a randomization process, it can be done on any observable characteristics, including outcomes, if they are available. The randomization should then record an indicator for each matching set, as these become equivalent to randomization strata and require controls in analysis. This approach is stratification taken to its most extreme: it reduces the number of potential randomizations dramatically from the possible number that would be available if the matching was not conducted, and therefore reduces the variance caused by the study design. When matching is done ex post in order to substitute for randomization, it is based on the assertion that within the matched groups, the assignment of treatment is as good as random. However, since most matching models rely on a specific linear model, such as **propensity score matching**,[145] they are open to the criticism of "specification searching", meaning that researchers can try different models of matching until one, by chance, leads to the final result that was desired; analytical approaches have shown that the better the fit of the matching model, the more likely it is that it has arisen by chance and is therefore biased.[146] Newer methods, such as **coarsened exact matching**,[147] are designed to remove some of the dependence on

[145] **Propensity Score Matching (PSM):** An estimation method that controls for the likelihood that each unit of observation would recieve treatment as predicted by observable characteristics.

[146] King, G. and Nielsen, R. (2019). Why propensity scores should not be used for matching. *Political Analysis*, 27(4):435–454

[147] Iacus, S. M., King, G., and Porro, G. (2012). Causal inference without balance checking: Coarsened exact matching. *Political Analysis*, 20(1):1–24

linearity. In all ex-post cases, pre-specification of the exact matching model can prevent some of the potential criticisms on this front, but ex-post matching in general is not regarded as a strong identification strategy.

Analysis of data from matching designs is relatively straightforward; the simplest design only requires controls (indicator variables) for each group or, in the case of propensity scoring and similar approaches, weighting the data appropriately in order to balance the analytical samples on the selected variables. The `teffects` suite in Stata provides a wide variety of estimators and analytical tools for various designs.[148] The coarsened exact matching (`cem`) package applies the nonparametric approach.[149] DIME's `iematch` command in the `ietoolkit` package produces matchings based on a single continuous matching variable.[150] In any of these cases, detailed reporting of the matching model is required, including the resulting effective weights of observations, since in some cases the lack of overlapping supports for treatment and control mean that a large number of observations will be weighted near zero and the estimated effect will be generated based on a subset of the data.

[148] https://ssc.wisc.edu/sscc/pubs/stata_psmatch.htm

[149] https://gking.harvard.edu/files/gking/files/cem-stata.pdf

[150] https://dimewiki.worldbank.org/wiki/iematch

Synthetic controls

Synthetic control is a relatively new method for the case when appropriate counterfactual individuals do not exist in reality and there are very few (often only one) treatment units.[151] For example, state- or national-level policy changes that can only be analyzed as a single unit are typically very difficult to find valid comparators for, since the set of potential comparators is usually small and diverse and therefore there are no close matches to the treated unit. Intuitively, the synthetic control method works by constructing a counterfactual version of the treated unit using an average of the other units available.[152] This is a particularly effective approach when the lower-level components of the units would be directly comparable: people, households, business, and so on in the case of states and countries; or passengers or cargo shipments in the case of transport corridors, for example.[153] This is because in those situations the average of the untreated units can be thought of as balancing by matching the composition of the treated unit.

To construct this estimator, the synthetic controls method requires retrospective data on the treatment unit and possible comparators, including historical data on the outcome of interest for all units. The counterfactual blend is chosen by optimizing the prediction of past outcomes based on the potential input characteristics, and typically selects a small set of comparators to weight into the final analysis.

[151] Abadie, A., Diamond, A., and Hainmueller, J. (2015). Comparative politics and the synthetic control method. *American Journal of Political Science*, 59(2):495–510

[152] Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic control methods for comparative case studies: Estimating the effect of california's tobacco control program. *Journal of the American statistical Association*, 105(490):493–505

[153] Gobillon, L. and Magnac, T. (2016). Regional policy evaluation: Interactive fixed effects and synthetic controls. *Review of Economics and Statistics*, 98(3):535–551

These datasets therefore may not have a large number of variables or observations, but the extent of the time series both before and after the implementation of the treatment are key sources of power for the estimate, as are the number of counterfactual units available. Visualizations are often excellent demonstrations of these results. The `synth` package provides functionality for use in Stata and R, although since there are a large number of possible parameters and implementations of the design it can be complex to operate.[154]

[154] https://web.stanford.edu/~jhain/synthpage.html

Chapter 4: Sampling, randomization, and power

Sampling and randomization are two core elements of research design. In experimental methods, sampling and randomization directly determine the set of individuals who are going to be observed and what their status will be for the purpose of effect estimation. Since we only get one chance to implement a given experiment, we need to have a detailed understanding of how these processes work and how to implement them properly. This allows us to ensure the field reality corresponds well to our experimental design. In quasi-experimental methods, sampling determines what populations the study will be able to make meaningful inferences about, and randomization analyses simulate counterfactual possibilities if the events being studied had happened differently. These analytical dimensions are particularly important in the initial phases of development research – typically conducted well before any actual fieldwork occurs – and often have implications for feasibility, planning, and budgeting.

Power calculations and randomization inference methods give us the tools to critically and quantitatively assess different sampling and randomization designs in light of our theories of change and to make optimal choices when planning studies. All random processes introduce statistical noise or uncertainty into the final estimates of effect sizes. Choosing one sample from all the possibilities produces some probability of choosing a group of units that are not, in fact, representative. Choosing one final randomization assignment similarly produces some probability of creating groups that are not good counterfactuals for each other. Power calculation and randomization inference are the main methods by which these probabilities of error are assessed. Good experimental design has high **power** – a low likelihood that statistical noise will substantially affect estimates of treatment effects.

Not all studies are capable of achieving traditionally high power: sufficiently precise sampling or treatment assignments may not be available. This may be especially true for novel or small-scale studies – things that have never been tried before may be hard to fund or execute at scale. What is important is that every study includes reasonable estimates of its power, so that the evidentiary value of its results can be assessed. Demonstrating that sampling and randomization were taken into consideration before going to field lends credibility to any research study. Using these tools to design the best experiments possible maximizes the likelihood that reported estimates are accurate.

Random processes in Stata

Most experimental designs rely directly on random processes, particularly sampling and randomization, to be executed in code. The fundamental econometrics behind impact evaluation depends on establishing that the sampling and treatment assignment processes are truly random. Therefore, understanding and correctly programming

for sampling and randomization is essential to ensuring that planned experiments are correctly implemented in the field, so that the results can be interpreted according to the experimental design. There are two distinct random processes referred to here: the conceptual process of assigning units to treatment arms, and the technical process of assigning random numbers in statistical software, which is a part of all tasks that include a random component.[155]

Randomization is challenging and its mechanics are unintuitive for the human brain. "True" randomization is also nearly impossible to achieve for computers, which are inherently deterministic.[156] For our purposes, we will focus on what you need to understand in order to produce truly random results for your project using Stata, and how you can make sure you can get those exact results again in the future. This takes a combination of strict rules, solid understanding, and careful programming. This section introduces the strict rules: these are non-negotiable (but thankfully simple). The second section provides basic introductions to the tasks of sampling and randomization, and the third introduces common varieties encountered in the field. The fourth section discusses more advanced topics that are used to analyze the random processes directly in order to understand their properties. However, field realities will inevitably be more complex than anything we present here, and you will need to recombine these lessons to match your project's needs.

[155] https://blog.stata.com/2016/03/10/how-to-generate-random-numbers-in-stata/

[156] https://www.random.org/randomness/

Reproducibility in random Stata processes

Any process that includes a random component is a random process, including sampling, randomization, power calculation simulations, and algorithms like bootstrapping. Reproducibility in statistical programming means that the outputs of random processes can be re-obtained at a future time. All random methods must be reproducible.[157] Stata, like most statistical software, uses a **pseudo-random number generator**. Basically, it has a really long ordered list of numbers with the property that knowing the previous one gives you precisely zero information about the next one. Stata uses one of these numbers every time it has a task that is non-deterministic. In ordinary use, it will cycle through these numbers starting from a fixed point every time you restart Stata, and by the time you get to any given script, the current state and the subsequent states will be as good as random.[158] However, for true reproducible randomization, we need two additional properties: we need to be able to fix the starting point so we can come back to it later; and we need to ensure that the starting point is independently random from our process. In Stata, this is accomplished through three command concepts:

[157] Orozco, V., Bontemps, C., Maigne, E., Piguet, V., Hofstetter, A., Lacroix, A., Levert, F., Rousselle, J.-M., et al. (2018). How to make a pie? reproducible research for empirical economics & econometrics. *Toulouse School of Economics Working Paper*, 933

[158] https://www.stata.com/manuals14/rsetseed.pdf

versioning, **sorting**, and **seeding**.

Versioning means using the same version of the software each time you run the random process. If anything is different, the underlying randomization algorithms may have changed, and it will be impossible to recover the original result. In Stata, the `version` command ensures that the software algorithm is fixed.[159] The `ieboilstart` command in `ietoolkit` provides functionality to support this requirement.[160] We recommend you use `ieboilstart` at the beginning of your master do-file.[161] However, testing your do-files without running them via the master do-file may produce different results, since Stata's `version` setting expires after each time you run your do-files.

Sorting means that the actual data that the random process is run on is fixed. Because numbers are assigned to each observation in row-by-row starting from the top row, changing their order will change the result of the process. A corollary is that the underlying data must be unchanged between runs: you must make a fixed final copy of the data when you run a randomization for fieldwork. In Stata, the only way to guarantee a unique sorting order is to use `isid [id_variable], sort`. (The `sort, stable` command is insufficient.) You can additionally use the `datasignature` command to make sure the data is unchanged.

Seeding means manually setting the start-point of the randomization algorithm. You can draw a uniformly distributed six-digit seed randomly by visiting `http://bit.ly/stata-random`. (This link is a just shortcut to request such a random seed on `https://www.random.org`.) There are many more seeds possible but this is a large enough set for most purposes. In Stata, `set seed [seed]` will set the generator to that state. You should use exactly one unique, different, and randomly created seed per randomization process. To be clear: you should not set a single seed once in the master do-file, but instead you should set a new seed in code right before each random process. The most important thing is that each of these seeds is truly random, so do not use shortcuts such as the current date or a seed you have used before. You will see in the code below that we include the source and timestamp for verification. Other commands may induce randomness in the data or alter the seed without you realizing it, so carefully confirm exactly how your code runs before finalizing it.[162]

To confirm that a randomization has worked well before finalizing its results, save the outputs of the process in a temporary location, re-run the code, and use `cf` or `datasignature` to ensure nothing has changed. It is also advisable to let someone else reproduce your randomization results on their machine to remove any doubt that

[159] At the time of writing we recommend using `version 13.1` for backward compatibility; the algorithm was changed after Stata 14 but its improvements do not matter in practice. You will *never* be able to reproduce a randomization in a different software, such as moving from Stata to R or vice versa.

[160] `https://dimewiki.worldbank.org/wiki/ieboilstart`

[161] `https://dimewiki.worldbank.org/wiki/Master_Do-files`

[162] `https://dimewiki.worldbank.org/wiki/Randomization_in_Stata`

your results are reproducable.

Sampling and randomization

The sampling and randomization processes we choose play an important role in determining the size of the confidence intervals for any estimates generated from that sample, and therefore our ability to draw conclusions. Random sampling and random assignment serve different purposes. Random *sampling* ensures that you have unbiased population estimates, and random *assignment* ensures that you have unbiased treatment estimates. If you randomly sample or assign a set number of observations from a set frame, there are a large – but fixed – number of permutations which you may draw.

In reality, you have to work with exactly one of them, so we put a lot of effort into making sure that one is a good one by reducing the probability that we observe nonexistent, or "spurious", results. In large studies, we can use what are called **asymptotic standard errors**[163] to express how far away from the true population parameters our estimates are likely to be. These standard errors can be calculated with only two datapoints: the sample size and the standard deviation of the value in the chosen sample. They are also typically the best case scenario for the population given the data structure. In small studies, such as those we often see in development, we have to be much more careful, particularly about practical considerations such as determining the representative population and fitting any constraints on study and sample design. This section introduces universal basic principles for sampling and randomization.

[163] Alecos Papadopoulos (https://stats.stackexchange.com/users/28746/alecos-papadopoulos). What is meant by the standard error of a maximum likelihood estimate? Cross Validated. https://stats.stackexchange.com/q/88491 (version: 2014-03-04)

Sampling

Sampling is the process of randomly selecting units of observation from a master list of individuals to be surveyed for data collection.[164] That master list may be called a **sampling universe**, a **listing frame**, or something similar. We recommend that this list be organized in a **master data set**[165], creating an authoritative source for the existence and fixed characteristics of each of the units that may be surveyed.[166] The master data set indicates how many individuals are eligible for data collection, and therefore contains statistical information about the likelihood that each will be chosen.

The simplest form of random sampling is **uniform-probability random sampling**. This means that every observation in the master data set has an equal probability of being included in the sample. The most explicit method of implementing this process is to assign random numbers to all your potential observations, order them by

[164] https://dimewiki.worldbank.org/wiki/Sampling_%26_Power_Calculations

[165] https://dimewiki.worldbank.org/wiki/Master_Data_Set

[166] https://dimewiki.worldbank.org/wiki/Unit_of_Observation

the number they are assigned, and mark as 'sampled' those with the lowest numbers, up to the desired proportion. (In general, we will talk about sampling proportions rather than numbers of observations. Sampling specific numbers of observations is complicated and should be avoided, because it will make the probability of selection very hard to calculate.) There are a number of shortcuts to doing this process, but they all use this method as the starting point, so you should become familiar with exactly how this method works.

Almost all of the relevant considerations for sampling come from two sources: deciding what population, if any, a sample is meant to represent (including subgroups); and deciding that different individuals should have different probabilities of being included in the sample. These should be determined in advance by the research design, since otherwise the sampling process will not be clear, and the interpretation of measurements is directly linked to who is included in them. Often, data collection can be designed to keep complications to a minimum, so long as it are carefully thought through from this perspective. Ex post changes to the study scope using a sample drawn for a different purpose usually involve tedious calculations of probabilities and should be avoided.

Randomization

Randomization, in this context, is the process of assigning units into treatment arms. Most of the code processses used for sampling are the same as those used for randomization, since randomization is also a process of splitting a sample into groups. Where sampling determines whether a particular individual will be observed at all in the course of data collection, randomization determines if each individual will be observed as a treatment unit or used as a counterfactual. Randomizing a treatment guarantees that, *on average*, the treatment will not be correlated with anything but the results of that treatment.[167] Causal inference from randomization therefore depends on a specific counterfactual: that the units who received the treatment program could just as well have been randomized into the control group. Therefore, controlling the exact probability that each individual receives treatment is the most important part of a randomization process, and must be carefully worked out in more complex designs.

[167] Duflo, E., Glennerster, R., and Kremer, M. (2007). Using randomization in development economics research: A toolkit. *Handbook of Development Economics*, 4:3895–3962

Just like sampling, the simplest form of randomization is a uniform-probability process.[168] Sampling typically has only two possible outcomes: observed and unobserved. Randomization, by contrast, often involves multiple possible results which each represent various varieties of treatments to be delivered; in some

[168] https://dimewiki.worldbank.org/wiki/Randomization_in_Stata

cases, multiple treatment assignments are intended to overlap in the same sample. Complexity can therefore grow very quickly in randomization and it is doubly important to fully understand the conceptual process that is described in the experimental design, and fill in any gaps in the process before implmenting it in Stata.

Some types of experimental designs necessitate that randomization results be revealed during data collection. It is possible to do this using survey software or live events. These methods typically do not leave a record of the randomization, so particularly when the experiment is done as part of data collection, it is best to execute the randomization in advance and preload the results. Even when randomization absolutely cannot be done in advance, it is still useful to build a corresponding model of the randomization process in Stata so that you can conduct statistical analysis later including checking for irregularities in the field assignment. Understanding that process will also improve the ability of the team to ensure that the field randomization process is appropriately designed and executed.

Clustering and stratification

For a variety of experimental and theoretical reasons, the actual sampling and randomization processes we need to perform are rarely as straightforward as a uniform-probability draw. We may only be able to implement treatment on a certain group of units (such as a school, a firm, or a market) or we may want to ensure that minority groups appear in either our sample or in specific treatment groups. The most common methods used in real studies are **clustering** and **stratification**. They allow us to control the randomization process with high precision, which is often necessary for appropriate inference, particularly when samples or subgroups are small.[169] These techniques can be used in any random process; their implementation is nearly identical in both sampling and randomization.

[169] Athey, S. and Imbens, G. W. (2017a). The econometrics of randomized experiments. *Handbook of Economic Field Experiments*, 1:73–140

Clustering

Many studies collect data at a different level of observation than the randomization unit.[170] For example, a policy may only be able to affect an entire village, but the study is interested in household behavior. This type of structure is called **clustering**,[171] and the groups in which units are assigned to treatment are called clusters. The same principle extends to sampling: it may be be necessary to observe all the children in a given teacher's classroom, for example.

[170] https://dimewiki.worldbank.org/wiki/Unit_of_Observation

[171] https://dimewiki.worldbank.org/wiki/Multi-stage_(Cluster)_Sampling

Clustering is procedurally straightforward in Stata, although it typically needs to be performed manually. To cluster a sampling or

randomization, create or use a data set where each cluster unit is an observation, randomize on that data set, and then merge back the results. When sampling or randomization is conducted using clusters, the clustering variable should be clearly identified since it will need to be used in subsequent statistical analysis. Namely, standard errors for these types of designs must be clustered at the level at which the randomization was clustered.[172] This accounts for the design covariance within the cluster – the information that if one individual was observed or treated there, the other members of the clustering group were as well.

[172] https://blogs.worldbank.org/impactevaluations/when-should-you-cluster-standard-errors-new-wisdom-econometrics-oracle

Stratification

Stratification is a research design component that breaks the full set of observations into a number of subgroups before performing randomization within each subgroup.[173] This has the effect of ensuring that members of each subgroup are included in all groups of the randomization process, since it is possible that a global randomization would put all the members of a subgroup into just one of the outcomes. In this context, the subgroups are called **strata**.

[173] https://dimewiki.worldbank.org/wiki/Stratified_Random_Sample

Manually implementing stratified randomization in Stata is prone to error. In particular, it is difficult to precisely account for the interaction of strata sizes with multiple treatment arms. Even for a very simple design, the method of randomly ordering the observations will often create very skewed assignments. This is especially true when a given stratum contains a small number of clusters, and when there are a large number of treatment arms, since the strata will rarely be exactly divisible by the number of arms.[174] The user-written `randtreat` command properly implements stratification. However, the options and outputs (including messages) from the command should be carefully reviewed so that you understand exactly what has been implemented. Notably, it is extremely hard to target precise numbers of observations in stratified designs, because exact allocations are rarely round fractions and the process of assigning the leftover "misfit" observations imposes an additional layer of randomization above the specified division.

[174] Carril, A. (2017). Dealing with misfits in random treatment assignment. *The Stata Journal*, 17(3):652–667

Whenever stratification is used for randomization, the analysis of differences within the strata (especially treatment effects) requires a control in the form of an indicator variable for all strata (fixed effects). This accounts for the fact that randomizations were conducted within the strata, comparing units to the others within its own strata by correcting for the local mean. Stratification is typically used for sampling in order to ensure that individuals with relevant characteristics will be observed; no adjustments are necessary as

long as the sampling proportion is constant across all strata. One common pitfall is to vary the sampling or randomization *probability* across different strata (such as "sample/treat all female heads of household"). If this is done, you must calculate and record the exact probability of inclusion for every unit, and re-weight observations accordingly.[175] The exact formula depends on the analysis being performed, but is usually related to the inverse of the likelihood of inclusion.

[175] http://blogs.worldbank.org/impactevaluations/tools-of-the-trade-when-to-use-those-sample-weights

Power calculation and randomization inference

Both sampling and randomization are noisy processes: they are random, after all, so it is impossible to predict the result in advance. By design, we know that the exact choice of sample or treatment will be uncorrelated with our key outcomes, but this lack of correlation is only true "in expectation" – that is, across a large number of randomizations. In any *particular* randomization, the correlation between the sampling or randomization and the outcome variable is guaranteed to be *nonzero*: this is called the **in-sample** or **finite-sample correlation**.

Since we know that the true correlation (over the "population" of potential samples or randomizations) is zero, we think of the observed correlation as an **error**. In sampling, we call this the **sampling error**, and it is defined as the difference between the true population parameter and the observed mean due to chance selection of units.[176] In randomization, we call this the **randomization noise**, and define it as the difference between the true treatment effect and the estimated effect due to placing units in groups. The intuition for both measures is that from any group, you can find some possible subsets that have higher-than-average values of some measure; similarly, you can find some that have lower-than-average values. Your sample or randomization will inevitably fall in one of these categories, and we need to assess the likelihood and magnitude of this occurence.[177] Power calculation and randomization inference are the two key tools to doing so.

[176] https://economistjourney.blogspot.com/2018/06/what-is-sampling-noise.html

[177] https://davegiles.blogspot.com/2019/04/what-is-permutation-test.html

Power calculations

Power calculations report the likelihood that your experimental design will be able to detect the treatment effects you are interested in. This measure of **power** can be described in various different ways, each of which has different practical uses.[178] The purpose of power calculations is to identify where the strengths and weaknesses of your design are located, so you know the relative tradeoffs

[178] http://www.stat.columbia.edu/~gelman/stuff_for_blog/chap20.pdf

you will face by changing your randomization scheme for the final design. They also allow realistic interpretations of evidence: results of low-power studies can be very interesting, but they have a correspondingly higher likelihood of reporting false positive results.

The classic definition of power is the likelihood that a design detects a significant treatment effect, given that there is a non-zero true effect in reality. This definition is useful retrospectively, but it can also be re-interpreted to help in experimental design. There are two common and useful practical applications of that definition that give actionable, quantitative results. The **minimum detectable effect (MDE)**[179] is the smallest true effect that a given research design can detect. This is useful as a check on whether a study is worthwhile. If, in your field, a "large" effect is just a few percentage points or a fraction of a standard deviation, then it is nonsensical to run a study whose MDE is much larger than that. This is because, given the sample size and variation in the population, the effect needs to be much larger to possibly be statistically detected, so such a study would not be able to say anything about the effect size that is practically relevant. Conversely, the **minimum sample size** pre-specifies expected effect sizes and tells you how large a study's sample would need to be to detect that effect, which can tell you what resources you would need to avoid that exact problem.

[179] https://dimewiki.worldbank.org/wiki/Minimum_Detectable_Effect

Stata has some commands that can calculate power analytically for very simple designs – `power` and `clustersampsi` – but they will not answer most of the practical questions that complex experimental designs require.[180] We suggest doing more advanced power calculations by simulation, since the interactions of experimental design, sampling and randomization, clustering, stratification, and treatment arms quickly becomes very complex. Furthermore, you should use real data on the population of interest whenever it is available, or you will have to make assumptions about the distribution of outcomes.

[180] https://dimewiki.worldbank.org/wiki/Power_Calculations_in_Stata

Together, the concepts of minimum detectable effect and minimum sample size can help answer a key question that typical approaches to power often do not. Namely, they can help you determine what trade-offs to make in the design of your experiment. Can you support another treatment arm? Is it better to add another cluster, or to sample more units per cluster? Simultaneously, planning out the regression structure in advance saves a lot of work once the data is in hand, and helps you think critically about what you are really testing. It also helps you to untangle design issues before they occur. Therefore, simulation-based power analysis is often more of a design aid than an output for reporting requirements. At the end of the day, you will probably have reduced the complexity of your experiment significantly. For reporting purposes, such as grant writing and

registered reports, simulation ensures you will have understood the key questions well enough to report standard measures of power once your design is decided.

Randomization inference

Randomization inference is used to analyze the likelihood that the randomization process, by chance, would have created a false treatment effect as large as the one you observed. Randomization inference is a generalization of placebo tests, because it considers what the estimated results would have been from a randomization that did not in fact happen in reality. Randomization inference is particularly important in quasi-experimental designs and in small samples, because these conditions usually lead to the situation where the number of possible *randomizations* is itself small. In those cases, we cannot rely on the usual assertion (a consequence of the Central Limit Theorem) that the variance of the treatment effect estimate is normal, and we therefore cannot use the "asymptotic" standard errors from Stata.

Instead, we directly simulate a large variety of possible alternative randomizations. Specifically, the user-written `ritest` command[181] allows us to execute a given randomization repeatedly, visualize how the randomization might have gone differently, and calculate empirical p-values for the effect size in our sample. After analyzing the actual treatment assignment, `ritest` illustrates the distribution of false correlations that this randomization approach can produce by chance between outcomes and treatments. The randomization-inference p-value is the number of times that a false effect was larger than the one you measured, and it is interpretable as the probability that a program with no effect would have given you a result like the one actually observed. These randomization inference[182] significance levels may be very different than those given by asymptotic confidence intervals, particularly in small samples (up to several hundred clusters).

[181] `http://hesss.org/ritest.pdf`

[182] `https://dimewiki.worldbank.org/wiki/Randomization_Inference`

Randomization inference can therefore be used proactively during experimental design. As long as there is some outcome data usable at this stage, you can use the same procedure to examine the potential treatment effects that your exact design is likely to produce. The range of these effects, again, may be very different from those predicted by standard approaches to power calculation, and randomization inference further allows visual inspection of results. If there is significant heaping at particular result levels, or if results seem to depend dramatically on the placement of a small number of individuals, randomization inference will flag those issues before

the experiment is fielded and allow adjustments to the design to be made.

replicability.do

```
* Set the version
    ieboilstart , v(13.1)
    `r(version)'

* Load the auto dataset and sort uniquely
    sysuse auto.dta , clear
    isid make, sort

* Set the seed using random.org (range: 100000 - 999999)
    set seed 287608 // Timestamp: 2019-02-17 23:06:36 UTC

* Demonstrate stability under the three rules
    gen check1 = rnormal()
    gen check2 = rnormal()

    set seed 287608
    gen check3 = rnormal()

//Visualize randomization results
    graph matrix check1 check2 check3 , half
```

randomization-cf.do

```
* Make one randomization
    sysuse bpwide.dta , clear
    isid patient, sort
    version 13.1
    set seed 796683 // Timestamp: 2019-02-26 22:14:17 UTC

    sample 100

* Save for comparison
    tempfile sample
    save    `sample' , replace

* Identical randomization
    sysuse bpwide.dta , clear
    isid patient, sort
    version 13.1
    set seed 796683 // Timestamp: 2019-02-26 22:14:17 UTC

    sample 100
    cf _all using `sample'

* Do something wrong
    sysuse bpwide.dta , clear
    sort bp*
    version 13.1
    set seed 796683 // Timestamp: 2019-02-26 22:14:17 UTC

    sample 100
    cf _all using `sample'
```

simple-sample.do

```
/*
    Simple reproducible sampling
*/

* Set up reproducbilitiy
    ieboilstart , v(12)     // Version
    `r(version)'            // Version
    sysuse auto.dta, clear  // Load data
    isid make, sort         // Sort
    set seed 215597         // Timestamp: 2019-04-26 17:51:02 UTC

* Take a sample of 20%
    preserve
        sample 20
        tempfile sample
        save    `sample' , replace
    restore

* Merge and complete
    merge 1:1 make using `sample'
    recode _merge (3 = 1 "Sampled") (* = 0 "Not Sampled") , gen(sample)
    label var sample "Sampled"
    drop _merge

* Check
    tab sample
```

sample-noise.do

```
* Reproducible setup: data, isid, version, seed
    sysuse auto.dta , clear
    isid make, sort
    version 13.1
    set seed 556292 // Timestamp: 2019-02-25 23:30:39 UTC

* Get true population parameter for price mean
    sum price
    local theMean = `r(mean)'

* Sample 20 units 1000 times and store the mean of [price]
    cap mat drop results              // Make matrix free
    qui forvalues i = 1/1000 {
        preserve
            sample 20 , count               // Remove count for 20%
            sum price                       // Calculate sample mean
            * Allow first run and append each estimate
            mat results = nullmat(results) \ [`r(mean)']
        restore
    }

* Load the results into memory and graph the distribution
    clear
    mat colnames results = "price_mean"
    svmat results , n(col)
    kdensity price_mean , norm xline(`theMean')
```

randomization-program-1.do

```
* Define a randomization program
cap prog drop my_randomization
    prog def  my_randomization

    * Syntax with open options for [ritest]
    syntax , [*]
    cap drop treatment

    * Group 2/5 in treatment and 3/5 in control
    xtile group = runiform() , n(5)
    recode group (1/2=0 "Control") (3/5=1 "Treatment") , gen(treatment)
    drop group

    * Cleanup
    lab var treatment "Treatment Arm"

end
```

randomization-program-2.do

```
* Reproducible setup: data, isid, version, seed
    sysuse auto.dta , clear
    isid make, sort
    version 13.1
    set seed 107738 // Timestamp: 2019-02-25 23:34:33 UTC

* Call the program
    my_randomization
    tab treatment

* Show randomization variation with [ritest]
    ritest treatment _b[treatment] ///
      , samplingprogram(my_randomization) kdensityplot ///
      : reg price treatment
```

randtreat-strata.do

```
* Use [randtreat] in randomization program ----------------
cap prog drop my_randomization
    prog def  my_randomization

    * Syntax with open options for [ritest]
    syntax , [*]
    cap drop treatment
    cap drop strata

    * Create strata indicator
    egen strata = group(sex agegrp) , label
        label var strata "Strata Group"

    * Group 1/5 in control and each treatment
    randtreat,              ///
        generate(treatment) /// New variable name
        multiple(6)         /// 6 arms
        strata(strata)      /// 6 strata
        misfits(global)     /// Randomized altogether

    * Cleanup
    lab var treatment "Treatment Arm"
    lab def treatment 0 "Control" 1 "Treatment 1" 2 "Treatment 2" ///
        3 "Treatment 3" 4 "Treatment 4" 5 "Treatment 5" , replace
    lab val treatment treatment
end // ---------------------------------------------------

* Reproducible setup: data, isid, version, seed
    sysuse bpwide.dta , clear
    isid patient, sort
    version 13.1
    set seed 796683 // Timestamp: 2019-02-26 22:14:17 UTC

* Randomize
   my_randomization
   tab treatment strata
```

randtreat-clusters.do

```
* Use [randtreat] in randomization program ----------------
cap prog drop my_randomization
    prog def  my_randomization

    * Syntax with open options for [ritest]
    syntax , [*]
    cap drop treatment
    cap drop cluster

    * Create cluster indicator
    egen cluster = group(sex agegrp) , label
    label var cluster "Cluster Group"

    * Save data set with all observations
    tempfile ctreat
    save  `ctreat' , replace

    * Keep only one from each cluster for randomization
    bysort cluster : keep if _n == 1

    * Group 1/2 in control and treatment in new variable treatment
    randtreat, generate(treatment) multiple(2)

    * Keep only treatment assignment and merge back to all observations
    keep cluster treatment
        merge 1:m cluster using `ctreat' , nogen

    * Cleanup
    lab var treatment "Treatment Arm"
    lab def treatment 0 "Control" 1 "Treatment" , replace
    lab val treatment treatment
end // ---------------------------------------------------

* Reproducible setup: data, isid, version, seed
    sysuse bpwide.dta , clear
    isid patient, sort
    version 13.1
    set seed 796683 // Timestamp: 2019-02-26 22:14:17 UTC

* Randomize
    my_randomization
    tab cluster treatment
```

minimum-detectable-effect.do

```
* Simulate power for different treatment effect sizes
   clear
   set matsize 5000
   cap mat drop results
   set seed 852526 // Timestamp: 2019-02-26 23:18:42 UTC

* Loop over treatment effect sizes (te)
* of 0 to 0.5 standard deviations
* in increments of 0.05 SDs
    qui forval te = 0(0.05)0.5 {
        forval i = 1/100 {      // Loop 100 times
            clear               // New simulation
            set obs 1000        // Set sample size to 1000

            * Randomly assign treatment
            * Here you could call a randomization program instead:
            gen t = (rnormal() > 0)

            * Simulate assumed effect sizes
            gen e = rnormal()       // Include a normal error term
            gen y = 1 + `te'*t + e  // Set functional form for DGP

            * Does regression detect an effect in this assignment?
            reg y t

            * Store the result
            mat a = r(table)                // Reg results
            mat a = a[....,1]               // t parameters
            mat results = nullmat(results) \ a' , [`te'] // First run and accumulate
        } // End iteration loop
    } // End incrementing effect size

* Load stored results into data
    clear
    svmat results , n(col)

* Analyze all the regressions we ran against power 80%
    gen sig = (pvalue <= 0.05) // Flag significant runs

* Proportion of significant results in each effect size group (80% power)
    graph bar sig , over(c10) yline(0.8)
```

minimum-sample-size.do

```
* Power for varying sample size & a fixed treatment effect
    clear
    set matsize 5000
    cap mat drop results
    set seed 510402 // Timestamp: 2019-02-26 23:19:00 UTC

* Loop over sample sizes (ss) 100 to 1000, increments of 100
    qui forval ss = 100(100)1000 {
        forval i = 1/100 {    // 100 iterations per each
            clear
            set obs `ss'      // Simulation with new sample size

            * Randomly assign treatment
            * Here you could call a randomization program instead
            gen t = (rnormal() > 0)

            * Simulate assumed effect size: here 0.2SD
            gen e = rnormal()     // Normal error term
            gen y = 1 + 0.2*t + e // Functional form for DGP

            * Does regression detect an effect in this assignment?
            reg y t

            * Store the result
            mat a = r(table)               // Reg results
            mat a = a[....,1]              // t parameters
            mat results = nullmat(results) \ a' , [`ss'] // First run and accumulate
        } // End iteration loop
    } // End incrementing sample size

* Load stored results into data
    clear
    svmat results , n(col)

* Analyze all the regressions we ran against power 80%
    gen sig = (pvalue <= 0.05) // Flag significant runs

* Proportion of significant results in each effect size group (80% power)
    graph bar sig , over(c10) yline(0.8)
```

Chapter 5: Collecting primary data

High quality research begins with a thoughtfully-designed, field-tested survey instrument, and a carefully supervised survey. Much of the recent push toward credibility in the social sciences has focused on analytical practices. We contest that credible research depends, first and foremost, on the quality of the raw data. This chapter covers the data generation workflow, from questionnaire design to field monitoring, for electronic data collection. There are many excellent resources on questionnaire design and field supervision, but few covering the particularly challenges and opportunities presented by electronic surveys. As there are many survey software, and the market is rapidly evolving, we focus on workflows and primary concepts, rather than software-specific tools. The chapter covers questionnaire design, piloting and programming; monitoring data quality during the survey; and how to ensure confidential data is handled securely from collection to storage and sharing.

Survey development workflow

A well-designed questionnaire results from careful planning, consideration of analysis and indicators, close review of existing questionnaires, survey pilots, and research team and stakeholder review. There are many excellent resources on questionnaire design, such as from the World Bank's Living Standards Measurement Survey. [183] The focus of this chapter is the particular design challenges for electronic surveys (often referred to as Computer Assisted Personal Interviews (CAPI)). [184]

Although most surveys are now collected electronically, by tablet, mobile phone or web browser, **questionnaire design**[185] (content development) and questionnaire programming (functionality development) should be seen as two strictly separate tasks. The research team should agree on all questionnaire content and design a paper version before programming an electronic version. This facilitates a focus on content during the design process and ensures teams have a readable, printable paper version of their questionnaire. Most importantly, it means the research, not the technology, drives the questionnaire design.

An easy-to-read paper version of the questionnaire is particularly critical for training enumerators, so they can get an overview of the survey content and structure before diving into the programming. It

[183] Grosh, Margaret; Glewwe, Paul. 2000. Designing Household Survey Questionnaires for Developing Countries : Lessons from 15 Years of the Living Standards Measurement Study, Volume 3. Washington, DC: World Bank. © World Bank.https://openknowledge.worldbank.org/handle/10986/15195License:CCBY3.0IGO.

[184] https://dimewiki.worldbank.org/wiki/Computer-Assisted_Personal_Interviews_(CAPI)

[185] https://dimewiki.worldbank.org/wiki/Questionnaire_Design

is much easier for enumerators to understand all possible response pathways from a paper version than from swiping question by question. Finalizing the questionnaire before programming also avoids version control concerns that arise from concurrent work on paper and electronic survey instruments. Finally, a paper questionnaire is an important documentation for data publication.

Content-focused Pilot

A **survey pilot**[186] is essential to finalize questionnaire design. A content-focused pilot[187] is best done on pen-and-paper, before the questionnaire is programmed. The objective is to improve the structure and length of the questionnaire, refine the phrasing and translation of specific questions, and confirm coded response options are exhaustive.[188] In addition, it is an opportunity to test and refine all survey protocols, such as how units will be sampled or pre-selected units identified. The pilot must be done out-of-sample, but in a context as similar as possible to the study sample.

[186] https://dimewiki.worldbank.org/wiki/Survey_Pilot

[187] https://dimewiki.worldbank.org/wiki/Piloting_Survey_Content

[188] https://dimewiki.worldbank.org/index.php?title=Checklist:_Refine_the_Questionnaire_(Content)&printable=yes

Data-focused pilot

A second survey pilot should be done after the questionnaire is programmed. The objective of the data-focused pilot[189] is to validate programming and export a sample dataset. Significant desk-testing of the instrument is required to debug the programming as fully as possible before going to the field. It is important to plan for multiple days of piloting, so that any further debugging or other revisions to the electronic survey instrument can be made at the end of each day and tested the following, until no further field errors arise. The data-focused pilot should be done in advance of enumerator training

[189] https://dimewiki.worldbank.org/index.php?title=Checklist:_Refine_the_Questionnaire_(Data)&printable=yes

Designing electronic questionnaires

The workflow for designing a questionnaire will feel much like writing an essay, or writing pseudocode: begin from broad concepts and slowly flesh out the specifics. It is essential to start with a clear understanding of the **theory of change**[190] and **experimental design** for your project. The first step of questionnaire design is to list key outcomes of interest, as well as the main factors to control for (covariates) and variables needed for experimental design. The ideal starting point for this is a **pre-analysis plan**.[191]

[190] https://dimewiki.worldbank.org/wiki/Theory_of_Change

[191] https://dimewiki.worldbank.org/wiki/Pre-Analysis_Plan

Use the list of key outcomes to create an outline of questionnaire *modules* (do not number the modules yet; instead use a short prefix so they can be easily reordered). For each module, determine if the module is applicable to the full sample, the appropriate respondent,

and whether or how often, the module should be repeated. A few examples: a module on maternal health only applies to household with a woman who has children, a household income module should be answered by the person responsible for household finances, and a module on agricultural production might be repeated for each crop the household cultivated.

Each module should then be expanded into specific indicators to observe in the field. [192] At this point, it is useful to do a **content-focused pilot** [193] of the questionnaire. Doing this pilot with a pen-and-paper questionnaire encourages more significant revisions, as there is no need to factor in costs of re-programming, and as a result improves the overall quality of the survey instrument.

[192] https://dimewiki.worldbank.org/wiki/Literature_Review_for_Questionnaire

[193] https://dimewiki.worldbank.org/wiki/Piloting_Survey_Content

Questionnaire design for quantitative analysis

This book covers surveys designed to yield datasets useful for quantitative analysis. This is a subset of surveys, and there are specific design considerations that will help to ensure the raw data outputs are ready for analysis.

From a data perspective, questions with pre-coded response options are always preferable to open-ended questions (the content-based pilot is an excellent time to ask open-ended questions, and refine responses for the final version of the questionnaire). Coding responses helps to ensure that the data will be useful for quantitative analysis. Two examples help illustrate the point. First, instead of asking "How do you feel about the proposed policy change?", use techniques like **Likert scales**[194]. Second, if collecting data on medication use or supplies, you could collect: the brand name of the product; the generic name of the product; the coded compound of the product; or the broad category to which each product belongs (antibiotic, etc.). All four may be useful for different reasons, but the latter two are likely to be the most useful for data analysis. The coded compound requires providing a translation dictionary to field staff, but enables automated rapid recoding for analysis with no loss of information. The generic class requires agreement on the broad categories of interest, but allows for much more comprehensible top-line statistics and data quality checks. Rigorous field testing is required to ensure that answer categories are comprehensive; however, it is best practice to include an *other, specify* option. Keep track of those reponses in the first few weeks of fieldwork; adding an answer category for a response frequently showing up as *other* can save time, as it avoids post-coding.

[194] **Likert scale:** an ordered selection of choices indicating the respondent's level of agreement or disagreement with a proposed statement.

It is useful to name the fields in your paper questionnaire in a way that will also work in the data analysis software. There is debate over

how individual questions should be identified: formats like `hq_1` are hard to remember and unpleasant to reorder, but formats like `hq_asked_about_loans` quickly become cumbersome. We recommend using descriptive names with clear prefixes so that variables within a module stay together when sorted alphabetically.[195] Variable names should never include spaces or mixed cases (all lower case is best). Take care with the length: very long names will be cut off in certain software, which could result in a loss of uniqueness. We discourage explicit question numbering, as it discourages re-ordering, which is a common recommended change after the pilot. In the case of follow-up surveys, numbering can quickly become convoluted, too often resulting in variables names like `ag_15a`, `ag_15_new`, `ag_15_fup2`, etc.

[195] https://medium.com/@janschenk/variable-names-in-survey-research-a18429d2d4d8

Questionnaires must include ways to document the reasons for **attrition**, treatment **contamination**, and **loss to follow-up**. These are essential data components for completing CONSORT records, a standardized system for reporting enrollment, intervention allocation, follow-up, and data analysis through the phases of a randomized trial of two groups.[196]

[196] Begg, C., Cho, M., Eastwood, S., Horton, R., Moher, D., Olkin, I., Pitkin, R., Rennie, D., Schulz, K. F., Simel, D., et al. (1996). Improving the quality of reporting of randomized controlled trials: The CONSORT statement. *JAMA*, 276(8):637–639

Once the content of the questionnaire is finalized and translated, it is time to proceed with programming the electronic survey instrument.

Programming electronic questionnaires

Electronic data collection has great potential to simplify survey implementation and improve data quality. Electronic questionnaires are typically created in a spreadsheet (e.g. Excel or Google Sheets) or software-specific form builder, accessible even to novice users. [197] We will not address software-specific form design in this book; rather, we focus on coding conventions that are important to follow for electronic surveys regardless of software choice. [198]

[197] https://dimewiki.worldbank.org/wiki/Questionnaire_Programming

[198] https://dimewiki.worldbank.org/wiki/SurveyCTO_Coding_Practices

Survey software tools provide a wide range of features designed to make implementing even highly complex surveys easy, scalable, and secure. However, these are not fully automatic: you still need to actively design and manage the survey. Here, we discuss specific practices that you need to follow to take advantage of electronic survey features and ensure that the exported data is compatible with the software that will be used for analysis.

As discussed above, the starting point for questionnare programming is a complete paper version of the questionnaire, piloted for content, and translated where needed. Doing so reduces version control issues that arise from making significant changes to concurrent paper and electronic survey instruments. When programming, do not start with the first question and proceed straight through to the last question. Instead, code from high level to small detail, following the

same questionnaire outline established at design phase. The outline provides the basis for pseudocode, allowing you to start with high level structure and work down to the level of individual questions. This will save time and reduce errors.

Electronic survey features

Electronic surveys are more than simply a paper questionnaire displayed on a mobile device or web browser. All common survey software allow you to automate survey logic and add in hard and soft constraints on survey responses. These features make enumerators' work easier, and they create the opportunity to identify and resolve data issues in real-time, simplifying data cleaning and improving response quality. Well-programmed questionnaires should include most or all of the following features:

- **Survey logic**: build in all logic, so that only relevant questions appear, rather than relying on enumerators to follow complex survey logic. This covers simple skip codes, as well as more complex interdependencies (e.g., a child health module is only asked to households that report the presence of a child under 5).
- **Range checks**: add range checks for all numeric variables to catch data entry mistakes (e.g. age must be less than 120).
- **Confirmation of key variables**: require double entry of essential information (such as a contact phone number in a survey with planned phone follow-ups), with automatic validation that the two entries match.
- **Multimedia**: electronic questionnaires facilitate collection of images, video, and geolocation data directly during the survey, using the camera and GPS built into the tablet or phone.
- **Preloaded data**: data from previous rounds or related surveys can be used to prepopulate certain sections of the questionnaire, and validated during the interview.
- **Filtered response options**: filters reduce the number of response options dynamically (e.g. filtering the cities list based on the state provided).
- **Location checks**: enumerators submit their actual location using in-built GPS, to confirm they are in the right place for the interview.
- **Consistency checks**: check that answers to related questions align, and trigger a warning if not so that enumerators can probe further (.e.g., if a household reports producing 800 kg of maize, but selling 900 kg of maize from their own production).

- **Calculations**: make the electronic survey instrument do all math, rather than relying on the enumerator or asking them to carry a calculator.

Compatibility with analysis software

All survey software include debugging and test options to correct syntax errors and make sure that the survey instruments will successfully compile. This is not sufficient, however, to ensure that the resulting dataset will load without errors in your data analysis software of choice. We developed the `ietestform` command,[199] part of the Stata package `iefieldkit`, to implement a form-checking routine for **SurveyCTO**, a proprietary implementation of **Open Data Kit (ODK)**. Intended for use during questionnaire programming and before fieldwork, `ietestform` tests for best practices in coding, naming and labeling, and choice lists. Although `ietestform` is software-specific, many of the tests it runs are general and important to consider regardless of software choice. To give a few examples, `ietestform` tests that no variable names exceed 32 characters, the limit in Stata (variable names that exceed that limit will be truncated, and as a result may no longer be unique). It checks whether ranges are included for numeric variables. `ietestform` also removes all leading and trailing blanks from response lists, which could be handled inconsistently across software.

[199] https://dimewiki.worldbank.org/wiki/ietestform

Data quality assurance

A huge advantage of electronic surveys, compared to traditional paper surveys, is the ability to access and analyze the data while the survey is ongoing. Data issues can be identified and resolved in real-time. Designing systematic data checks, and running them routinely throughout data collection, simplifies monitoring and improves data quality. As part of survey preparation, the research team should develop a **data quality assurance plan** [200]. While data collection is ongoing, a research assistant or data analyst should work closely with the field team to ensure that the survey is progressing correctly, and perform **high-frequency checks (HFCs)** of the incoming data. [201] Data quality assurance requires a combination of real-time data checks and survey audits. Careful field supervision is also essential for a successful survey; however, we focus on the first two in this chapter, as they are the most directly data related.

[200] https://dimewiki.worldbank.org/wiki/Data_Quality_Assurance_Plan

[201] https://github.com/PovertyAction/high-frequency-checks/wiki

High frequency checks

High-frequency checks (HFCs) should carefully inspect key treatment and outcome variables so that the data quality of core experimental variables is uniformly high, and that additional field effort is centered where it is most important. Data quality checks should be run on the data every time it is downloaded (ideally on a daily basis), to flag irregularities in survey progress, sample completeness or response quality. `ipacheck`[202] is a very useful command that automates some of these tasks.

[202] https://github.com/PovertyAction/high-frequency-checks

It is important to check every day that the units interviewed match the survey sample. Many survey software include case management features, through which sampled units are directly assigned to individual enumerators. Even with careful management, it is often the case that raw data includes duplicate entries, which may occur due to field errors or duplicated submissions to the server. Even with careful management, it is often the case that raw data includes duplicate entries, which may occur due to field errors or duplicated submissions to the server.[203] `ieduplicates`[204] provides a workflow for collaborating on the resolution of duplicate entries between you and the field team. Next, observed units in the data must be validated against the expected sample: this is as straightforward as merging the sample list with the survey data and checking for mismatches. Reporting errors and duplicate observations in real-time allows the field team to make corrections efficiently. Tracking survey progress is important for monitoring attrition, so that it is clear early on if a change in protocols or additional tracking will be needed. It is also important to check interview completion rate and sample compliance by surveyor and survey team, to identify any under-performing individuals or teams.

[203] https://dimewiki.worldbank.org/wiki/Duplicates_and_Survey_Logs

[204] https://dimewiki.worldbank.org/wiki/ieduplicates

When all data collection is complete, the survey team should prepare a final field report, which should report reasons for any deviations between the original sample and the dataset collected. Identification and reporting of **missing data** and **attrition** is critical to the interpretation of survey data. It is important to structure this reporting in a way that not only groups broad rationales into specific categories but also collects all the detailed, open-ended responses to questions the field team can provide for any observations that they were unable to complete. This reporting should be validated and saved alongside the final raw data, and treated the same way. This information should be stored as a dataset in its own right – a **tracking dataset** – that records all events in which survey substitutions and loss to follow-up occurred in the field and how they were implemented and resolved.

High frequency checks should also include survey-specific data checks. As electronic survey software incorporates many data control features, discussed above, these checks should focus on issues survey software cannot check automatically. As most of these checks are survey specific, it is difficult to provide general guidance. An in-depth knowledge of the questionnaire, and a careful examination of the pre-analysis plan, is the best preparation. Examples include consistency across multiple responses, complex calculation (such as crop yield, which first requires unit conversions), suspicious patterns in survey timing, or atypical response patters from specific enumerators. timing, or atypical response patterns from specific enumerators.[205] survey software typically provides rich metadata, which can be useful in assessing interview quality. For example, automatically collected time stamps show how long enumerators spent per question, and trace histories show how many times answers were changed before the survey was submitted.

[205] https://dimewiki.worldbank.org/wiki/Monitoring_Data_Quality

High-frequency checks will only improve data quality if the issues they catch are communicated to the field. There are lots of ways to do this; what's most important is to find a way to create actionable information for your team, given field constraints. 'ipacheck' generates an excel sheet with results for each run; these can be sent directly to the field teams. Many teams choose other formats to display results, notably online dashboards created by custom scripts. It is also possible to automate communication of errors to the field team by adding scripts to link the HFCs with a messaging program such as whatsapp. Any of these solutions are possible: what works best for your team will depend on such variables as cellular networks in fieldwork areas, whether field supervisors have access to laptops, internet speed, and coding skills of the team preparing the HFC workflows.

Data considerations for field monitoring

Careful monitoring of field work is essential for high quality data. **Back-checks**[206] and other survey audits help ensure that enumerators are following established protocols, and are not falsifying data. For back-checks, a random subset of the field sample is chosen and a subset of information from the full survey is verified through a brief interview with the original respondent. Design of the back-check questionnaire follows the same survey design principles discussed above: you should use the pre-analysis plan or list of key outcomes to establish which subset of variables to prioritize.

[206] https://dimewiki.worldbank.org/wiki/Back_Checks

Real-time access to the survey data increases the potential utility of back-checks dramatically, and both simplifies and improves the

rigor of related workflows. You can use the raw data to draw the back-check sample; assuring it is appropriately apportioned across interviews and survey teams. As soon as back-checks are complete, the back-check data can be tested against the original data to identify areas of concern in real-time. `bcstats` is a useful tool for analyzing back-check data in Stata module. [207]

[207] https://ideas.repec.org/c/boc/bocode/s458173.html

Electronic surveys also provide a unique opportunity to do audits through audio recordings of the interview, typically short recordings triggered at random throughout the questionnaire. **Audio audits** are a useful means to assess whether the enumerator is conducting the interview as expected (and not sitting under a tree making up data). Do note, however, that audio audits must be included in the Informed Consent.

Collecting Data Securely

Primary data collection almost always includes **personally-identifiable information (PII)** [208]. PII must be handled with great care at all points in the data collection and management process, to comply with ethical requirements and avoid breaches of confidentiality. Access to PII must be restricted to team members granted that permission by the applicable Institutional Review Board or a data licensing agreement with a partner agency. Research teams must maintain strict protocols for data security at each stage of the process: data collection, storage, and sharing.

[208] https://dimewiki.worldbank.org/wiki/Personally_Identifiable_Information_(PII)

Secure data in the field

All mainstream data collection software automatically **encrypt** [209] all data submitted from the field while in transit (i.e., upload or download). Your data will be encrypted from the time it leaves the device (in tablet-assisted data collation) or your browser (in web data collection), until it reaches the server. Therefore, as long as you are using an established survey software, this step is largely taken care of. However, the research team must ensure that all computers, tablets, and accounts used have a logon password and are never left unlocked.

[209] **Encryption:** the process of making information unreadable to anyone without access to a specific deciphering key. https://dimewiki.worldbank.org/wiki/Encryption

Secure data storage

Encryption at rest is the only way to ensure that PII data remains private when it is stored on someone else's server on the internet. You must keep your data encrypted on the server whenever PII data is collected. Encryption makes data files completely unusable without access to a security key specific to that data – a higher level

of security than password-protection. Encryption at rest requires active participation from the user, and you should be fully aware that if your private key is lost, there is absolutely no way to recover your data.

You should not assume that your data is encrypted by default: indeed, for most survey software platforms, encryption needs to be enabled by the user. To enable it, you must confirm you know how to operate the encryption system and understand the consequences if basic protocols are not followed. When you enable encryption, the service will allow you to download – once – the keyfile pair needed to decrypt the data. You must download and store this in a secure location, such as a password manager. Make sure you store keyfiles with descriptive names to match the survey to which they correspond. Any time anyone accesses the data- either when viewing it in the browser or downloading it to your computer- they will be asked to provide the keyfile. Only project team members named in the IRB are allowed access to the private keyfile.

To proceed with data analysis, you typically need a working copy of the data accessible from a personal computer. The following workflow allows you to receive data from the server and store it securely, without compromising data security.

1. Download data

2. Store a "master" copy of the data into an encrypted location that will remain accessible on disk and be regularly backed up

3. Create a "gold master" copy of the raw data in a secure location, such as a long-term cloud storage service or an encrypted physical hard drive stored in a separate location. If you remain lucky, you will never have to access this copy – you just want to know it is out there, safe, if you need it.

This handling satisfies the rule of three: there are two on-site copies of the data and one off-site copy, so the data can never be lost in case of hardware failure. In addition, you should ensure that all teams take basic precautions to ensure the security of data, as most problems are due to human error. Ideally, the machine hard drives themselves should also be encrypted, as well as any external hard drives or flash drives used. All files sent to the field containing PII data, such as sampling lists, must be encrypted. You must never share passwords by email; rather, use a secure password manager. This significantly mitigates the risk in case there is a security breach such as loss, theft, hacking, or a virus, with little impact on day-to-day utilization.

Secure data sharing

To simplify workflow, it is best to remove PII variables from your data at the earliest possible opportunity, and save a de-identified copy of the data. Once the data is de-identified, it no longer needs to be encrypted - therefore you can interact with it directly, without having to provide the keyfile.

We recommend de-identification in two stages: an initial process to remove direct identifiers to create a working de-identified dataset, and a final process to remove all possible identifiers to create a publishable dataset. The **initial de-identification** should happen directly after the encrypted data is downloaded to disk. At this time, for each variable that contains PII, ask: will this variable be needed for analysis? If not, the variable should be dropped. Examples include respondent names, enumerator names, interview date, respondent phone number. If the variable is needed for analysis, ask: can I encode or otherwise construct a variable to use for the analysis that masks the PII, and drop the original variable? Examples include geocoordinates (after constructing measures of distance or area, drop the specific location), and names for social network analysis (can be encoded to unique numeric IDs). If PII variables are directly required for the analysis itself, it will be necessary to keep at least a subset of the data encrypted through the data analysis process.

Flagging all potentially identifying variables in the questionnaire design stage, as recommended above, simplifies the initial de-identification. You already have the list of variables to assess, and ideally have already assessed those against the pre-analysis plan. If so, all you need to do is write a script to drop the variables that are not required for analysis, encode or otherwise mask those that are required, and save a working version of the data.

The **final de-identification** is a more involved process, with the objective of creating a dataset for publication that cannot be manipulated or linked to identify any individual research participant. You must remove all direct and indirect identifiers, and assess the risk of statistical disclosure. [210]. There will almost always be a trade-off between accuracy and privacy. For publicly disclosed data, you should favor privacy. There are a number of useful tools for de-identification: PII scanners for Stata [211] or R [212], and tools for statistical disclosure control. [213] In cases where PII data is required for analysis, we recommend embargoing the sensitive variables when publishing the data. Access to the embargoed data could be granted for the purposes of study replication, if approved by an IRB.

With the raw data securely stored and backed up, and a de-identified dataset to work with, you are ready to move to data

[210] Disclosure risk: the likelihood that a released data record can be associated with an individual or organization

[211] https://github.com/J-PAL/stata_PII_scan

[212] https://github.com/J-PAL/PII-Scan

[213] https://sdcpractice.readthedocs.io/en/latest/

cleaning, and analysis.

Chapter 6: Analyzing survey data

Transforming raw data into a substantial contribution to scientific knowledge requires a mix of subject expertise, programming skills, and statistical and econometric knowledge. The process of data analysis is, therefore, a back-and-forth discussion between people with differing skill sets. The research assistant usually ends up being the pivot of this discussion. It is their job to translate the data received from the field into economically meaningful indicators and to analyze them while making sure that code and outputs do not become too difficult to follow or get lost over time.

When it comes to code, though, analysis is the easy part, *as long as you have organized your data well*. Of course, there is plenty of complexity behind it: the econometrics, the theory of change, the measurement methods, and so much more. But none of those are the subject of this book. *Instead, this chapter will focus on how to organize your data work so that coding the analysis becomes easy*. Most of a Research Assistant's time is spent cleaning data and getting it into the right format. When the practices recommended here are adopted, analyzing the data is as simple as using a command that is already implemented in a statistical software.

Data management

The goal of data management is to organize the components of data work so it can traced back and revised without massive effort. In our experience, there are four key elements to good data management: folder structure, task breakdown, master scripts, and version control. A good folder structure organizes files so that any material can be found when needed. It reflects a task breakdown into steps with well-defined inputs, tasks, and outputs. This breakdown is applied to code, data sets, and outputs. A master script connects folder structure and code. It is a one-file summary of your whole project. Finally, version histories and backups enable the team to edit files without fear of losing information. Smart use of version control also allows you to track how each edit affects other files in the project.

Folder structure

There are many schemes to organize research data. Our preferred scheme reflects the task breakdown just discussed. DIME Analytics created the `iefolder`[214] package (part of `ietoolkit`[215]) to standardize

[214] https://dimewiki.worldbank.org/wiki/iefolder

[215] https://dimewiki.worldbank.org/wiki/ietoolkit

folder structures across teams and projects. This means that PIs and RAs face very small costs when switching between projects, because they are organized in the same way.[216] We created the command based on our experience with primary data, but it can be used for different types of data. Whatever you team may need in terms of organization, the principle of creating one standard remains.

[216] https://dimewiki.worldbank.org/wiki/DataWork_Folder

At the first level of the structure created by `iefolder` are what we call survey round folders.[217] You can think of a "round" as one source of data, that will be cleaned in the same script. Inside round folders, there are dedicated folders for raw (encrypted) data; de-identified data; cleaned data; and final (constructed) data. There is a folder for raw results, as well as for final outputs. The folders that hold code are organized in parallel to these, so that the progression through the whole project can be followed by anyone new to the team. Additionally, `iefolder` creates **master do-files**[218] so all project code is reflected in a top-level script.

[217] https://dimewiki.worldbank.org/wiki/DataWork_Survey_Round

[218] https://dimewiki.worldbank.org/wiki/Master_Do-files

Task breakdown

We divide the process of turning raw data into analysis data into three stages: data cleaning, variable construction, and data analysis. Though they are frequently implemented at the same time, we find that creating separate scripts and data sets prevents mistakes. It will be easier to understand this division as we discuss what each stage comprises. What you should know by now is that each of these stages has well-defined inputs and outputs. This makes it easier to track tasks across scripts, and avoids duplication of code that could lead to inconsistent results. For each stage, there should be a code folder and a corresponding data set. The names of codes, data sets and outputs for each stage should be consistent, making clear how they relate to one another. So, for example, a script called `clean-section-1` would create a data set called `cleaned-section-1`.

The division of a project in stages also helps the review workflow inside your team. The code, data and outputs of each of these stages should go through at least one round of code review. During the code review process, team members should read and run each other's codes. Doing this at the end of each stage helps prevent the amount of work to be reviewed to become too overwhelming. Code review is a common quality assurance practice among data scientists. It helps to keep the level of the outputs high, and is also a great way to learn and improve your code.

Master scripts

Master scripts allow users to execute all the project code from a single file. They briefly describes what each code, and maps the files they require and create. They also connects code and folder structure through globals or objects. In short, a master script is a human-readable map to the tasks, files and folder structure that comprise a project. Having a master script eliminates the need for complex instructions to replicate results. Reading the master do-file should be enough for anyone unfamiliar with the project to understand what are the main tasks, which scripts execute them, and where different files can be found in the project folder. That is, it should contain all the information needed to interact with a project's data work.

Version control

Finally, everything that can be version-controlled should be. Version control allows you to effectively track code edits, including the addition and deletion of files. This way you can delete code you no longer need, and still recover it easily if you ever need to get back previous work. Both analysis results and data sets will change with the code. You should have each of them stored with the code that created it. If you are writing code in Git/GitHub, you can output plain text files such as `.tex` tables and metadata saved in `.txt` or `.csv` to that directory. Binary files that compile the tables, as well as the complete data sets, on the other hand, should be stored in your team's shared folder. Whenever data cleaning or data construction codes are edited, use the master script to run all the code for your project. Git will highlight the changes that were in data sets and results that they entail.

Data cleaning

Data cleaning is the first stage of transforming the data you received from the field into data that you can analyze.[219] The cleaning process involves (1) making the data set easily usable and understandable, and (2) documenting individual data points and patterns that may bias the analysis. The underlying data structure does not change. The cleaned data set should contain only the variables collected in the field. No modifications to data points are made at this stage, except for corrections of mistaken entries.

Cleaning is probably the most time consuming of the stages discussed in this chapter. This is the time when you obtain an extensive understanding of the contents and structure of the data that was collected. Explore your data set using tabulations, summaries,

[219] https://dimewiki.worldbank.org/wiki/Data_Cleaning

and descriptive plots. You should use this time to understand the types of responses collected, both within each survey question and across respondents. Knowing your data set well will make it possible to do analysis.

De-identification

The initial input for data cleaning is the raw data. It should contain only materials that are received directly from the field. They will invariably come in a host of file formats and nearly always contain personally-identifying information. These files should be retained in the raw data folder *exactly as they were received*. Be mindful of where this file is stored. Maintain a backup copy in a secure offsite location. Every other file is created from the raw data, and therefore can be recreated. The exception, of course, is the raw data itself, so it should never be edited directly. The rare and only case when the raw data can be edited directly is when it is encoded incorrectly and some non-English character is causing rows or columns to break at the wrong place when the data is imported. In this scenario, you will have to remove the special character manually, save the resulting data set *in a new file* and securely back up *both* the broken and the fixed version of the raw data.

Note that no one who is not listed in the IRB should be able to access its content, not even the company providing file-sharing services. Check if your organization has guidelines on how to store data securely, as they may offer an institutional solution. If that is not the case, you will need to encrypt the data, especially before sharing it, and make sure that only IRB-listed team members have the encryption key.[220].

Secure storage of the raw data means access to it will be restricted even inside the research team.[221] Loading encrypted data frequently can be disruptive to the workflow. To facilitate the handling of the data, remove any personally identifiable information from the data set. This will create a de-identified data set, that can be saved in a non-encrypted folder. De-identification,[222] at this stage, means stripping the data set of direct identifiers such as names, phone numbers, addresses, and geolocations.[223] The resulting de-identified data will be the underlying source for all cleaned and constructed data. Because identifying information is typically only used during data collection, to find and confirm the identity of interviewees, de-identification should not affect the usability of the data. In fact, most identifying information can be converted into non-identified variables for analysis purposes (e.g. GPS coordinates can be translated into distances). However, if sensitive information is strictly needed for

[220] https://dimewiki.worldbank.org/wiki/Encryption

[221] https://dimewiki.worldbank.org/wiki/Data_Security

[222] https://dimewiki.worldbank.org/wiki/De-identification

[223] https://www.povertyactionlab.org/sites/default/files/resources/J-PAL-guide-to-deidentifying-data.pdf

analysis, the data must be encrypted while performing the tasks described in this chapter.

Correction of data entry errors

There are two main cases when the raw data will be modified during data cleaning. The first one is when there are duplicated entries in the data. Ensuring that observations are uniquely and fully identified[224] is possibly the most important step in data cleaning. Modern survey tools create unique observation identifiers. That, however, is not the same as having a unique ID variable for each individual in the sample. You want to make sure the data set has a unique ID variable that can be cross-referenced with other records, such as the Master Data Set[225] and other rounds of data collection. `ieduplicates` and `iecompdup`, two Stata commands included in the `iefieldkit` package,[226] create an automated workflow to identify, correct and document occurrences of duplicate entries.

[224] https://dimewiki.worldbank.org/wiki/ID_Variable_Properties

[225] https://dimewiki.worldbank.org/wiki/Master_Data_Set

[226] https://dimewiki.worldbank.org/wiki/iefieldkit

Looking for duplicated entries is usually part of data quality monitoring, as is the only other reason to change the raw data during cleaning: correcting mistakes in data entry. During data quality monitoring, you will inevitably encounter data entry mistakes, such as typos and inconsistent values. These mistakes should be fixed in the cleaned data set, and you should keep a careful record of how they were identified, and how the correct value was obtained.

Labeling and annotating the raw data

On average, making corrections to primary data is more time-consuming than when using secondary data. But you should always check for possible issues in any data you are about to use. The last step of data cleaning, however, will most likely still be necessary. It consists of labeling and annotating the data, so that its users have all the information needed to interact with it. This is a key step to making the data easy to use, but it can be quite repetitive. The `iecodebook` command suite, also part of `iefieldkit`, is designed to make some of the most tedious components of this process, such as renaming, relabeling, and value labeling, much easier.[227] We have a few recommendations on how to use this command for data cleaning. First, we suggest keeping the same variable names in the cleaned data set as in the survey instrument, so it's straightforward to link data points for a variable to the question that originated them. Second, don't skip the labeling. Applying labels makes it easier to understand what the data is showing while exploring the data. This minimizes the risk of small errors making their way through into the analysis stage. Variable and value labels should be accurate and

[227] https://dimewiki.worldbank.org/wiki/iecodebook

concise.[228] Third, recodes should be used to turn codes for "Don't know", "Refused to answer", and other non-responses into extended missing values.[229] String variables need to be encoded, and open-ended responses, categorized or dropped[230] (unless you are using qualitative or classification analyses, which are less common). Finally, any additional information collected only for quality monitoring purposes, such as notes and duration fields, can also be dropped.

[228] https://dimewiki.worldbank.org/wiki/Data_Cleaning#Applying_Labels

[229] https://dimewiki.worldbank.org/wiki/Data_Cleaning#Survey_Codes_and_Missing_Values

[230] https://dimewiki.worldbank.org/wiki/Data_Cleaning#Strings

Documenting data cleaning

Throughout the data cleaning process, you will need inputs from the field, including enumerator manuals, survey instruments, supervisor notes, and data quality monitoring reports. These materials are essential for data documentation.[231] They should be stored in the corresponding "Documentation" folder for easy access, as you will probably need them during analysis, and they must be made available for publication. Include in the `Documentation` folder records of any corrections made to the data, including to duplicated entries, as well as communications from the field where theses issues are reported. Be very careful not to include sensitive information in documentation that is not securely stored, or that you intend to release as part of a replication package or data publication.

[231] https://dimewiki.worldbank.org/wiki/Data_Documentation

Another important component of data cleaning documentation are the results of As clean your data set, take the time to explore the variables in it. Use tabulations, histograms and density plots to understand the structure of data, and look for potentially problematic patterns such as outliers, missing values and distributions that may be caused by data entry errors. Don't spend time trying to correct data points that were not flagged during data quality monitoring. Instead, create a record of what you observe, then use it as a basis to discuss with your team how to address potential issues during data construction. This material will also be valuable during exploratory data analysis.

The cleaned data set

The main output of data cleaning is the cleaned data set. It should contain the same information as the raw data set, with no changes to data points. It should also be easily traced back to the survey instrument, and be accompanied by a dictionary or codebook. Typically, one cleaned data set will be created for each data source, i.e. per survey instrument. Each row in the cleaned data set represents one survey entry or unit of observation.[232] If the raw data set is very large, or the survey instrument is very complex, you may want to break the data cleaning into sub-steps, and create intermediate

[232] Wickham, H. (2014). Tidy data. *The Journal of Statistical Software*, 59

cleaned data sets (for example, one per survey module). Breaking cleaned data sets into the smallest unit of observation inside a roster make the cleaning faster and the data easier to handle during construction. But having a single cleaned data set will help you with sharing and publishing the data. To make sure this file doesn't get too big to be handled, use commands such as `compress` in Stata to make sure the data is always stored in the most efficient format. Once you have a cleaned, de-identified data set, and documentation to support it, you have created the first data output of your project: a publishable data set. The next chapter will get into the details of data publication. For now, all you need to know is that your team should consider submitting the data set for publication at this point, even if it will remain embargoed for some time. This will help you organize your files and create a back up of the data, and some donors require that the data be filed as an intermediate step of the project.

Indicator construction

The second stage in the creation of analysis data is construction. Constructing variables means processing the data points as provided in the raw data to make them suitable for analysis. It is at this stage that the raw data is transformed into analysis data. This is done by creating derived variables (dummies, indices, and interactions, to name a few), as planned during research design, and using the pre-analysis plan as a guide. To understand why construction is necessary, let's take the example of a household survey's consumption module. For each item in a context-specific bundle, it will ask whether the household consumed any of it over a certain period of time. If they did, it will then ask about quantities, units and expenditure for each item. However, it is difficult to run a meaningful regression on the number of cups of milk and handfuls of beans that a household consumed over a week. You need to manipulate them into something that has *economic* meaning, such as caloric input or food expenditure per adult equivalent. During this process, the data points will typically be reshaped and aggregated so that level of the data set goes from the unit of observation (one item in the bundle) in the survey to the unit of analysis (the household).[233]

[233] https://dimewiki.worldbank.org/wiki/Unit_of_Observation

Why construction?

Construction is done separately from data cleaning for two reasons. The first one is to clearly differentiate the data originally collected from the result of data processing decisions. The second is to ensure that variable definition is consistent across data sources. Unlike

cleaning, construction can create many outputs from many inputs. Let's take the example of a project that has a baseline and an endline survey. Unless the two instruments are exactly the same, which is preferable but often not the case, the data cleaning for them will require different steps, and therefore will be done separately. However, you still want the constructed variables to be calculated in the same way, so they are comparable. To do this, you will at least two cleaning scripts, and a single one for construction – we will discuss how to do this in practice in a bit.

Ideally, indicator construction should be done right after data cleaning, according to the pre-analysis plan. In practice, however, following this principle is not always easy. As you analyze the data, different constructed variables will become necessary, as well as subsets and other alterations to the data. Still, constructing variables in a separate script from the analysis will help you ensure consistency across different outputs. If every script that creates a table starts by loading a data set, subsetting it and manipulating variables, any edits to construction need to be replicated in all scripts. This increases the chances that at least one of them will have a different sample or variable definition. Therefore, even if construction ends up coming before analysis only in the order the code is run, it's important to think of them as different steps.

Construction tasks and how to approach them

The first thing that comes to mind when we talk about variable construction is, of course, creating new variables. Do this by adding new variables to the data set instead of overwriting the original information, and assign functional names to them. During cleaning, you want to keep all variables consistent with the survey instrument. But constructed variables were not present in the survey to start with, so making their names consistent with the survey form is not as crucial. Of course, whenever possible, having variable names that are both intuitive *and* can be linked to the survey is ideal, but if you need to choose, prioritize functionality. Ordering the data set so that related variables are together and adding notes to each of them as necessary will also make your data set more user-friendly.

The most simple case of new variables to be created are aggregate indicators. For example, you may want to add a household's income from different sources into a single total income variable, or create a dummy for having at least one child in school. Jumping to the step where you actually create this variables seems intuitive, but it can also cause you a lot of problems, as overlooking details may affect your results. It is important to check and double-check the value-

assignments of questions and their scales before constructing new variables based on them. This is when you will use the knowledge of the data you acquired and the documentation you created during the cleaning step the most. It is often useful to start looking at comparisons and other documentation outside the code editor.

Make sure to standardize units and recode categorical variables so their values are consistent. It's possible that your questionnaire asked respondents to report some answers as percentages and others as proportions, or that in one variable 0 means "no" and 1 means "yes", while in another one the same answers were coded are 1 and 2. We recommend coding yes/no questions as either 1/0 or TRUE/FALSE, so they can be used numerically as frequencies in means and as dummies in regressions. Check that non-binary categorical variables have the same value-assignment, i.e., that labels and levels have the same correspondence across variables that use the same options. Finally, make sure that any numeric variables you are comparing are converted to the same scale or unit of measure. You cannot add one hectare and twos acres into a meaningful number.

During construction, you will also need to address some of the issues you identified in the data during data cleaning. The most common of them is the presence of outliers. How to treat outliers is a research question, but make sure to note what we the decision made by the research team, and how you came to it. Results can be sensitive to the treatment of outliers, so keeping the original variable in the data set will allow you to test how much it affects the estimates. All these points also apply to imputation of missing values and other distributional patterns.

The more complex construction tasks involve changing the structure of the data: adding new observations or variables by merging data sets, and changing the unit of observation through collapses or reshapes. There are always ways for things to go wrong that we never anticipated, but two issues to pay extra attention to are missing values and dropped observations. Merging, reshaping and aggregating data sets can change both the total number of observations and the number of observations with missing values. Make sure to read about how each command treats missing observations and, whenever possible, add automated checks in the script that throw an error message if the result is changing. If you are subsetting your data, drop observations explicitly, indicating why you are doing that and how the data set changed.

Finally, primary panel data involves additional timing complexities. It is common to construct indicators soon after receiving data from a new survey round. However, creating indicators for each round separately increases the risk of using different definitions every

time. Having a well-established definition for each constructed variable helps prevent that mistake, but the best way to guarantee it won't happen is to create the indicators for all rounds in the same script. Say you constructed variables after baseline, and are now receiving midline data. Then the first thing you should do is create a panel data set – {textttiecodebook's `append` subcommand will help you reconcile and append survey rounds. After that, adapt the construction code so it can be used on the panel data set. Apart from preventing inconsistencies, this process will also save you time and give you an opportunity to review your original code.

Documenting indicators construction

Because data construction involves translating concrete data points to more abstract measurements, it is important to document exactly how each variable is derived or calculated. Adding comments to the code explaining what you are doing and why is a crucial step both to prevent mistakes and to guarantee transparency. To make sure that these comments can be more easily navigated, it is wise to start writing a variable dictionary as soon as you begin making changes to the data. Carefully record how specific variables have been combined, recoded, and scaled, and refer to those records in the code. This can be part of a wider discussion with your team about creating protocols for variable definition, which will guarantee that indicators are defined consistently across projects. When all your final variables have been created, you can use `iecodebook`'s `export` subcommand to list all variables in the data set, and complement it with the variable definitions you wrote during construction to create a concise meta data document. Documentation is an output of construction as relevant as the code and the data. Someone unfamiliar with the project should be able to understand the contents of the analysis data sets, the steps taken to create them, and the decision-making process through your documentation. The construction documentation will complement the reports and notes created during data cleaning. Together, they will form a detailed account of the data processing.

Constructed data sets

The other set of construction outputs, as expected, consists of the data sets that will be used for analysis. A constructed data set is built to answer an analysis question. Since different pieces of analysis may require different samples, or even different units of observation, you may have one or multiple constructed data sets, depending on how your analysis is structured. So don't worry if you cannot create a single, "canonical" analysis data set. It is common to have many

purpose-built analysis datasets. Think of an agricultural intervention that was randomized across villages and only affected certain plots within each village. The research team may want to run household-level regressions on income, test for plot-level productivity gains, and check if village characteristics are balanced. Having three separate datasets for each of these three pieces of analysis will result in much cleaner do files than if they all started from the same file.

Writing data analysis code

Data analysis is the stage when research outputs are created. Many introductions to common code skills and analytical frameworks exist, such as *R for Data Science*;[234] *A Practical Introduction to Stata*;[235] *Mostly Harmless Econometrics*;[236] and *Causal Inference: The Mixtape*.[237] This section will not include instructions on how to conduct specific analyses. That is a research question, and requires expertise beyond the scope of this book. Instead, we will outline the structure of writing analysis code, assuming you have completed the process of data cleaning and construction.

[234] https://r4ds.had.co.nz/

[235] https://scholar.harvard.edu/files/mcgovern/files/practical_introduction_to_stata.pdf

[236] https://www.researchgate.net/publication/51992844_Mostly_Harmless_Econometrics_An_Empiricist's_Companion

[237] http://scunning.com/mixtape.html

Organizing analysis code

The analysis stage usually starts with a process we call exploratory data analysis. This is when you are trying different things and looking for patterns in your data. It progresses into final analysis when your team starts to decide what are the main results, those that will make it into the research output. The way you deal with code and outputs for exploratory and final analysis is different, and this section will discuss how. During exploratory data analysis, you will be tempted to write lots of analysis into one big, impressive, start-to-finish script. It subtly encourages poor practices such as not clearing the workspace and not reloading the constructed data set before each analysis task. It's important to take the time to organize scripts in a clean manner and to avoid mistakes.

A well-organized analysis script starts with a completely fresh workspace and explicitly loads data before analyzing it. This encourages data manipulation to be done earlier in the workflow (that is, during construction). It also and prevents you from accidentally writing pieces of analysis code that depend on one another and requires manual instructions for all required code snippets be run in the right order. Each script should run completely independently of all other code. You can go as far as coding every output in a separate script. There is nothing wrong with code files being short and simple – as long as they directly correspond to specific pieces of analysis.

Analysis files should be as simple as possible, so whoever is reading it can focus on the econometrics. All research decisions should be made very explicit in the code. This includes clustering, sampling, and control variables, to name a few. If you have multiple analysis data sets, each of them should have a descriptive name about its sample and unit of observation. As your team comes to a decision about model specification, you can create globals or objects in the master script to use across scripts. This is a good way to make sure specifications are consistent throughout the analysis. Using pre-specified globals or objects also makes your code more dynamic, so it is easy to update specifications and results without changing every script. It is completely acceptable to have folders for each task, and compartmentalize each analysis as much as needed.

Exporting outputs

To accomplish this, you will need to make sure that you have an effective data management system, including naming, file organization, and version control. Just like you did with each of the analysis datasets, name each of the individual analysis files descriptively. Code files such as `spatial-diff-in-diff.do`, `matching-villages.R`, and `summary-statistics.py` are clear indicators of what each file is doing, and allow you to find code quickly. If you intend to numerically order the code as they appear in a paper or report, leave this to near publication time.

Whole books have been written on how to create good data visualizations, so we will not attempt to give you advice on it. Rather, here are a few resources we have found useful. The Tapestry conference focuses on "storytelling with data".[238] *Fundamentals of Data Visualization* provides extensive details on practical application;[239] as does *Data Visualization: A Practical Introduction*.[240] Graphics tools like Stata are highly customizable. There is a fair amount of learning curve associated with extremely-fine-grained adjustment, but it is well worth reviewing the graphics manual[241] For an easier way around it, Gray Kimbrough's *Uncluttered Stata Graphs* code is an excellent default replacement for Stata graphics that is easy to install. [242] If you are a R user, the *R Graphics Cookbook*[243] is a great resource for the most popular visualization package `ggplot`[244]. But there are a variety of other visualization packages, such as `highcharter`[245], `r2d3`[246], `leaflet`[247], and `plotly`[248], to name a few. We have no intention of creating an exhaustive list, and this one is certainly missing very good references. But at least it is a place to start.

[238] https://www.youtube.com/playlist?list=PLb0GkPPcZCVE9EAm9qhlg5eXMgLrrfMRq

[239] https://serialmentor.com/dataviz

[240] http://socvis.co

[241] https://www.stata.com/manuals/g.pdf

[242] https://graykimbrough.github.io/uncluttered-stata-graphs/

[243] https://r-graphics.org/

[244] https://ggplot2.tidyverse.org/

[245] http://jkunst.com/highcharter/

[246] https://rstudio.github.io/r2d3/

[247] https://rstudio.github.io/leaflet/

[248] https://plot.ly/r/

Exporting analysis outputs

Our team has created a few products to automate common outputs and save you precious research time. The `ietoolkit` package includes two commands to export nicely formatted tables. `iebaltab`[249] creates and exports balance tables to excel or LaTeX. `ieddtab`[250] does the same for difference-in-differences regressions. The **Stata Visual Library**[251] has examples of graphs created in Stata and curated by us.[252] **Data visualization**[253] is increasingly popular, and is becoming a field in its own right.[254] We attribute some of this to the difficulty of writing code to create them. Making a visually compelling graph would already be hard enough if you didn't have to go through many rounds of googling to understand a command. The trickiest part of using plot commands is to get the data in the right format. This is why the **Stata Visual Library** includes example data sets to use with each do-file.

It's ok to not export each and every table and graph created during exploratory analysis. Final analysis scripts, on the other hand, should export final outputs, which are ready to be included to a paper or report. No manual edits, including formatting, should be necessary after exporting final outputs – those that require copying and pasting edited outputs, in particular, are absolutely not advisable. Manual edits are difficult to replicate, and you will inevitably need to make changes to the outputs. Automating them will save you time by the end of the process. However, don't spend too much time formatting tables and graphs until you are ready to publish.[255] Polishing final outputs can be a time-consuming process, and you want to it as few times as possible.

We cannot stress this enough: don't ever set a workflow that requires copying and pasting results. Copying results from excel to word is error-prone and inefficient. Copying results from a software console is risk-prone, even more inefficient, and unnecessary. There are numerous commands to export outputs from both R and Stata to a myriad of formats.[256] Save outputs in accessible and, whenever possible, lightweight formats. Accessible means that it's easy for other people to open them. In Stata, that would mean always using `graph export` to save images as `.jpg`, `.png`, `.pdf`, etc., instead of `graph save`, which creates a `.gph` file that can only be opened through a Stata installation. Some publications require "lossless" TIFF of EPS files, which are created by specifying the desired extension. Whichever format you decide to use, remember to always specify the file extension explicitly. For tables there are less options and more consideration to be made. Exporting table to `.tex` should be preferred. Excel `.xlsx` and `.csv` are also commonly used,

[249] https://dimewiki.worldbank.org/wiki/Iebaltab

[250] https://dimewiki.worldbank.org/wiki/Ieddtab

[251] https://worldbank.github.io/Stata-IE-Visual-Library/

[252] A similar resource for R is *The R Graph Gallery*. https://www.r-graph-gallery.com/

[253] https://dimewiki.worldbank.org/wiki/Data_visualization

[254] Healy, K. (2018). *Data visualization: A practical introduction*. Princeton University Press; and Wilke, C. O. (2019). *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. O'Reilly Media

[255] For a more detailed discussion on this, including different ways to export tables from Stata, see https://github.com/bbdaniels/stata-tables

[256] Some examples are `estout`, `outreg2`, and `outwrite` in Stata, and `stargazer` and `ggsave` in R.

but require the extra step of copying the tables into the final output. The amount of work needed in a copy-paste workflow increases rapidly with the number of tables and figures included in a research output, and do the chances of having the wrong version a result in your paper or report.

If you need to create a table with a very particular format, that is not automated by any command you know, consider writing the it manually (Stata's `filewrite`, for example, allows you to do that). This will allow you to write a cleaner script that focuses on the econometrics, and not on complicated commands to create and append intermediate matrices. To avoid cluttering your scripts with formatting and ensure that formatting is consistent across outputs, define formatting options in an R object or a Stata global and call them when needed. Keep in mind that final outputs should be self-standing. This means it should be easy to read and understand them with only the information they contain. Make sure labels and notes cover all relevant information, such as sample, unit of observation, unit of measurement and variable definition.[257]

If you follow the steps outlined in this chapter, most of the data work involved in the last step of the research process – publication – will already be done. If you used de-identified data for analysis, publishing the cleaned data set in a trusted repository will allow you to cite your data. Some of the documentation produced during cleaning and construction can be published even if your data is too sensitive to be published. Your analysis code will be organized in a reproducible way, so will need to do release a replication package is a last round of code review. This will allow you to focus on what matters: writing up your results into a compelling story.

[257] https://dimewiki.worldbank.org/wiki/Checklist:_Reviewing_Graphs https://dimewiki.worldbank.org/wiki/Checklist:_Submit_Table

Chapter 7: Publishing collaborative research

Publishing academic research today extends well beyond writing up and submitting a Word document alone. Typically, various contributors collaborate on both code and writing, manuscripts go through many iterations and revisions, and the final package for publication includes not just a manuscript but also the code and data used to generate the results. Ideally, your team will spend as little time as possible fussing with the technical requirements of publication. It is in nobody's interest for a skilled and busy researcher to spend days re-numbering references (and it can take days) if a small amount of up-front effort could automate the task. In this section we suggest several methods – collectively refered to as "dynamic documents" – for managing the process of collaboration on any technical product.

For most research projects, completing a manuscript is not the end of the task. Academic journals increasingly require submission of a replication package, which contains the code and materials needed to create the results. These represent an intellectual contribution in their own right, because they enable others to learn from your process and better understand the results you have obtained. Holding code and data to the same standards a written work is a new practice for many researchers. In this chapter, we provide guidelines that will help you prepare a functioning and informative replication package. In all cases, we note that technology is rapidly evolving and that the specific tools noted here may not remain cutting-edge, but the core principles involved in publication and transparency will endure.

Collaborating on technical writing

It is increasingly rare that a single author will prepare an entire manuscript alone. More often than not, documents will pass back and forth between several writers before they are ready for publication, so it is essential to use technology and workflows that avoid conflicts. Just as with the preparation of analytical outputs, this means adopting tools and practices that enable tasks such as version control and simultaneous contribution. Furthermore, it means preparing documents that are **dynamic** – meaning that updates to the analytical outputs that constitute them can be passed on to the final output with a single process, rather than copy-and-pasted or otherwise handled individually. Thinking of the writing process in this way is intended to improve organization and reduce error, such that there is no risk of materials being compiled with out-of-date results, or of completed work being lost or redundant.

Dynamic documents

Dynamic documents are a broad class of tools that enable a streamlined, reproducible workflow. The term "dynamic" can refer to any document-creation technology that allows the inclusion of explicitly encoded linkages to raw output files. This means that, whenever outputs are updated, the next time the document is loaded or compiled, it will automatically include all changes made to all outputs without any additional intervention from the user. This means that updates will never be accidentally excluded, and it further means that updating results will not become more difficult as the number of inputs grows, because they are all managed by a single integrated process.

You will note that this is not possible in tools like Microsoft Office, although there are various tools and add-ons that produce similar functionality, and we will introduce some later in this book. In Word, by default, you have to copy and paste each object individually whenever tables, graphs, or other inputs have to be updated. This creates complex inefficiency: updates may be accidentally excluded and ensuring they are not will become more difficult as the document grows. As time goes on, it therefore becomes more and more likely that a mistake will be made or something will be missed. Therefore this is a broadly unsuitable way to prepare technical documents.

There are a number of tools that can be used for dynamic documents. Some are code-based tools such as R's RMarkdown[258] and Stata's `dyndoc`[259]. These tools "knit" or "weave" text and code together, and are programmed to insert code outputs in pre-specified locations. Documents called "notebooks" (such as Jupyter[260]) work similarly, as they also use the underlying analytical software to create the document. These types of dynamic documents are usually appropriate for short or informal materials because they tend to offer restricted editability outside the base software and often have limited abilities to incorporate precision formatting.

[258] https://rmarkdown.rstudio.com/

[259] https://www.stata.com/manuals/rptdyndoc.pdf

[260] https://jupyter.org/

There are other dynamic document tools which do not require direct operation of the underlying code or software, simply access to the updated outputs. These can be useful for working on informal outputs, such as blogposts, with collaborators who do not code. An example of this is Dropbox Paper, a free online writing tool that allows linkages to files in Dropbox, which are automatically updated anytime the file is replaced.

However, the most widely utilized software for dynamically managing both text and results is LaTeX (pronounced "lah-tek").[261] Rather than using a coding language that is built for another purpose or trying to hide the code entirely, LaTeX is a special code language

[261] https://github.com/worldbank/DIME-LaTeX-Templates

designed for document preparation and typesetting. While this tool has a significant learning curve, its enormous flexibility in terms of operation, collaboration, and output formatting and styling makes it the primary choice for most large technical outputs today, and it has proven to have enduring popularity. In fact, LaTeX operates behind-the-scenes in many of the tools listed before. Therefore, we recommend that you learn to use LaTeX directly as soon as you are able to and provide several resources for doing so.

Technical writing with LaTeX

LaTeX is billed as a "document preparation system". What this means is worth unpacking. In LaTeX, instead of writing in a "what-you-see-is-what-you-get" mode as you do in Word or the equivalent, you write plain text interlaced with coded instructions for formatting (similar in concept to HTML). Because it is written in a plain text file format, `.tex` can be version-controlled using Git. This is why it has become the dominant "document preparation system" in technical writing. LaTeX enables automatically-organized documents, manages tables and figures dynamically, and includes commands for simple markup like font styles, paragraph formatting, section headers and the like. It also includes special controls for including tables and figures, footnotes and endnotes, complex mathematical notation, and automated bibliography preparation. It also allows publishers to apply global styles and templates to already-written material, allowing them to reformat entire documents in house styles with only a few keystrokes.

One of the most important tools available in LaTeX is the BibTeX bibliography manager.[262] BibTeX keeps all the references you might use in an auxiliary file, then references them using a simple element typed directly in the document: a `cite` command. The same principles that apply to figures and tables are therefore applied here: You can make changes to the references in one place (the `.bib` file), and then everywhere they are used they are updated correctly with one process. Specifically, LaTeX inserts references in text using the `\cite{}` command. Once this is written, LaTeX automatically pulls all the citations into text and creates a complete bibliography based on the citations you use when you compile the document. The system allows you to specify exactly how references should be displayed in text (such as superscripts, inline references, etc.) as well as how the bibliography should be styled and in what order (such as Chicago, MLA, Harvard, or other common styles). This tool is so widely used that it is natively integrated in Google Scholar. To obtain a reference in the `.bib` format for any paper you find, click "BibTeX" at the

[262] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.365.3194&rep=rep1&type=pdf

bottom of the Cite window (below the preformatted options). Then, copy the code directly from Google Scholar into your `.bib` file. They will look like the following:

sample.bib

```
@article{flom2005latex,
  title={LATEX for academics and researchers who (think they) don't need it},
  author={Flom, Peter},
  journal={The PracTEX Journal},
  volume={4},
  year={2005},
  publisher={Citeseer}
}
```

BibTeX citations are then used as follows:

citation.tex

```
With these tools, you can ensure that references are handled
in a format you can manage and control.\cite{flom2005latex}
```

With these tools, you can ensure that references are handled in a format you can manage and control.[263] Finally, LaTeX has one more useful trick: using **pandoc**,[264] you can translate the raw document into Word (or a number of other formats) by running the following code from the command line:

[263] Flom, P. (2005). LaTeX for academics and researchers who (think they) don't need it. *The PracTEX Journal*, 4

[264] http://pandoc.org/

pandoc.sh

```
pandoc -s -o main.docx main.tex --bibliography sample.bib --csl=[style].csl
```

The last portion after `csl=` specifies the bibliography style. You can download a CSL (Citation Styles Library) file[265] for nearly any journal and have it applied automatically in this process. Therefore, even in the case where you are requested to provide `.docx` versions of materials to others, or tracked-changes versions, you can create them effortlessly, and use external tools like Word's compare feature to generate integrated tracked versions when needed.

[265] https://github.com/citation-style-language/styles

Unfortunately, despite these advantages, LaTeX can be a challenge to set up and use at first, particularly if you are new to working with plain text code and file management. It is also unfortunately weak with spelling and grammar checking. This is because LaTeX requires that all formatting be done in its special code language, and it is not particularly informative when you do something wrong. This can be off-putting very quickly for people who simply want to get to writing, like senior researchers. While integrated editing

and compiling tools like TeXStudio[266] and `atom-latex`[267] offer the most flexibility to work with LaTeX on your computer, such as advanced integration with Git, the entire group of writers needs to be comfortable with LaTeX before adopting one of these tools. They can require a lot of troubleshooting at a basic level at first, and non-technical staff may not be willing or able to acquire the required knowledge. Therefore, to take advantage of the features of LaTeX, while making it easy and accessible to the entire writing team, we need to abstract away from the technical details where possible.

[266] https://www.texstudio.org

[267] https://atom.io/packages/atom-latex

Getting started with LaTeX in the cloud

LaTeX is a challenging tool to get started using, but the control it offers over the writing process is invaluable. In order to make it as easy as possible for your team to use LaTeX without all members having to invest in new skills, we suggest using a web-based implementation as your first foray into LaTeX writing. Most such sites offer a subscription feature with useful extensions and various sharing permissions, and some offer free-to-use versions with basic tools that are sufficient for a broad variety of applications, up to and including writing a complete academic paper with coauthors.

Cloud-based implementations of LaTeX are suggested here for several reasons. Since they are completely hosted online, they avoids the inevitable troubleshooting of setting up a LaTeX installation on various personal computers run by the different members of your team. They also typically maintain a single continuously synced master copy of the document so that different writers do not create conflicted or out-of-sync copies, or need to deal with Git themselves to maintain that sync. They typically allow inviting collaborators to edit in a fashion similar to Google Docs, though different services vary the number of collaborators and documents allowed at each tier. Most importantly, some tools provide a "rich text" editor that behaves pretty similarly to familiar tools like Word, so that collaborators can write text directly into the document without worrying too much about the underlying LaTeX coding. Cloud services also usually offer a convenient selection of templates so it is easy to start up a project and see results right away without needing to know a lot of the code that controls document formatting.

On the downside, there is a small amount of up-front learning required, continous access to the Internet is necessary, and updating figures and tables requires a bulk file upload that is tough to automate. One of the most common issues you will face using online editors will be special characters which, because of code functions, need to be handled differently than in Word. Most critically, the ampersand

(&), percent (%), and underscore (_) need to be "escaped" (interpreted as text and not code) in order to render. This is done by by writing a backslash (\) before them, such as writing `40\%` for the percent sign to appear in text. Despite this, we believe that with minimal learning and workflow adjustments, cloud-based implementations are often the easiest way to allow coauthors to write and edit in LaTeX so long as you make sure you are available to troubleshoot minor issues like these.

Preparing a complete replication package

While we have focused so far on the preparation of written materials for publication, it is increasingly important for you to consider how you will publish the data and code you used for your research as well. More and more major journals are requiring that publications provide direct links to both the code and data used to create the results, and some even require being able to reproduce the results themselves before they will approve a paper for publication.[268] If your material has been well-structured throughout the analytical process, this will only require a small amount of extra work; if not, paring it down to the "replication package" may take some time. A complete replication package should accomplish several core functions. It must provide the exact data and code that is used for a paper, all necessary de-identified data for the analysis, and all code necessary for the analysis. The code should exactly reproduce the raw outputs you have used for the paper, and should not include documentation or data you would not share publicly. This usually means removing project-related documentation such as contracts and details of data collection and other field work, and double-checking all datasets for potentially identifying information.

[268] https://www.aeaweb.org/journals/policies/data-code/

Publishing data for replication

Enabling permanent access to the data used in your study is an important contribution you can make along with the publication of results. It allows other researchers to validate the mechanical construction of your results, to investigate what other results might be obtained from the same population, and test alternative approaches to other questions. Therefore you should make clear in your study where and how data are stored, and how and under what circumstances it might be accessed. You do not always have to publish the data yourself, and in some cases you are legally not allowed to, but what matters is that the data is published (with or without access restrictions) and that you cite or otherwise directly reference all

data, even data that you cannot release. When your raw data is owned by someone else, or for any other reason you are not able to publish it, in many cases you will have the right to release at least some subset of your constructed data set, even if it is just the derived indicators you constructed. If you have questions about your rights over original or derived materials, check with the legal team at your organization or at the data provider's. You should only directly publish data which is fully de-identified and, to the extent required to ensure reasonable privacy, potentially identifying characteristics are further masked or removed. In all other cases, you should contact an appropriate data catalog to determine what privacy and licensing options are available.

Make sure you have a clear understanding of the rights associated with the data release and communicate them to any future users of the data. You must provide a license with any data release.[269] This document need not be extremely detailed, but it should clearly communicate to the reader what they are allowed to do with your data and how credit should be given and to whom in further work that uses it. Keep in mind that you may or may not own your data, depending on how it was collected, and the best time to resolve any questions about these rights is at the time that data collection or transfer agreements are signed. Even if you cannot release data immediately or publicly, there are often options to catalog or archive the data without open publication. These may take the form of metadata catalogs or embargoed releases. Such setups allow you to hold an archival version of your data which your publication can reference, as well as provide information about the contents of the datasets and how future users might request permission to access them (even if you are not the person who can grant that permission). They can also provide for timed future releases of datasets once the need for exclusive access has ended.

[269] https://iatistandard.org/en/guidance/preparing-organisation/organisation-data-publication/how-to-license-your-data/

Data publication should release the dataset in a widely recognized format. While software-specific datasets are acceptable accompaniments to the code (since those precise materials are probably necessary), you should also consider releasing generic datasets such as CSV files with accompanying codebooks, since these will be re-adaptable by any researcher. Additionally, you should also release the data collection instrument or survey questionnaire so that readers can understand which data components are collected directly in the field and which are derived. You should publish both a clean version of the data which corresponds exactly to the original database or questionnaire as well as the constructed or derived dataset used for analysis. Wherever possible, you should also release the code that constructs any derived measures, particularly where definitions may

vary, so that others can learn from your work and adapt it as they like.

Publishing code for replication

Before publishing your code, you should edit it for content and clarity just as if it were written material. The purpose of releasing code is to allow others to understand exactly what you have done in order to obtain your results, as well as to apply similar methods in future projects. Therefore it should both be functional and readable. If you've followed the recommendations in this book, this will be much easier to do. Code is often not written this way when it is first prepared, so it is important for you to review the content and organization so that a new reader can figure out what and how your code should do. Therefore, whereas your data should already be very clean at this stage, your code is much less likely to be so, and this is where you need to make time investments prior to releasing your replication package. By contrast, replication code usually has few legal and privacy constraints. In most cases code will not contain identifying information; but make sure to check carefully that it does not. Publishing code also requires assigning a license to it; in a majority of cases, code publishers like GitHub offer extremely permissive licensing options by default. (If you do not provide a license, nobody can use your code!)

Make sure the code functions identically on a fresh install of your chosen software. A new user should have no problem getting the code to execute perfectly. In either a scripts folder or in the root directory, include a master script (dofile or R script for example). The master script should allow the reviewer to change a single line of code: the one setting the directory path. After that, running the master script should run the entire project and re-create all the raw outputs exactly as supplied. Indicate the filename and line to change. Check that all your code will run completely on a new computer: Install any required user-written commands in the master script (for example, in Stata using `ssc install` or `net install` and in R include code giving users the option to install packages, including selecting a specific version of the package if necessary). In many cases you can even directly provide the underlying code for any user-installed packages that are needed to ensure forward-compatibility. Make sure system settings like `version`, `matsize`, and `varabbrev` are set.

Finally, make sure that the code and its inputs and outputs are clearly identified. A new user should, for example, be able to easily identify and remove any files created by the code so that they can be recreated quickly. They should also be able to quickly map all

the outputs of the code to the locations where they are placed in the associated published material, such as ensuring that the raw components of figures or tables are clearly identified. Documentation in the master script is often used to indicate this information. For example, outputs should clearly correspond by name to an exhibit in the paper, and vice versa. (Supplying a compiling LaTeX document can support this.) Code and outputs which are not used should be removed.

Releasing a replication package

If you are at this stage, all you need to do is find a place to publish your materials. This is slightly easier said than done, as there are a few variables to take into consideration and, at the time of writing, no global consensus on the best solution. The technologies available are likely to change dramatically over the next few years; the specific solutions we mention here highlight some current approaches as well as their strengths and weaknesses. GitHub provides one solution. Making a GitHub repository public is completely free. It can hold any file types, provide a structured download of your whole project, and allow others to look at alternate versions or histories easily. It is straightforward to simply upload a fixed directory to GitHub apply a sharing license, and obtain a URL for the whole package. (However, there is a strict size restriction of 100MB per file and a restriction on the size of the repository as a whole, so larger projects will need alternative solutions.)

However, GitHub is not ideal for other reasons. It is not built to hold data in an efficient way or to manage licenses or citations for datasets. It does not provide a true archive service – you can change or remove the contents at any time. A repository such as the Harvard Dataverse[270] addresses these issues, as it is designed to be a citable data repository; the IPA/J-PAL field experiment repository is especially relevant.[271] The Open Science Framework[272] can also hold both code and data, as can ResearchGate.[273] Some of these will also assign a permanent digital object identifier (DOI) link for your work. Any of these locations is acceptable – the main requirement is that the system can handle the structured directory that you are submitting, and that it can provide a stable, structured URL for your project and report exactly what, if any, modifications you have made since initial publication. You can even combine more than one tool if you prefer, as long as they clearly point to each other. Emerging technologies such as the "containerization" approach of Docker or CodeOcean[274] offer to store both code and data, and also provide an online workspace in which others can execute and modify your

[270] https://dataverse.harvard.edu

[271] https://www.povertyactionlab.org/blog/9-11-19/new-hub-data-randomized-evaluations

[272] https://osf.io

[273] https://https://www.researchgate.net

[274] https://codeocean.com

code without having to download your tools and match your local environment when packages and other underlying software may have changed since publication.

In addition to code and data, you may also want to release an author's copy or preprint of the article itself along with these raw materials. Check with your publisher before doing so; not all journals will accept material that has been released. Therefore you may need to wait until acceptance is confirmed. This can be done on a number of preprint websites, many of which are topic-specific.[275] You can also use GitHub and link to the PDF file directly on your personal website or whatever medium you are sharing the preprint through. Do not use Dropbox or Google Drive for this purpose: many organizations do not allow access to these tools, and that includes blocking staff from accessing your material.

[275] https://en.wikipedia.org/wiki/ArXiv

Appendix: The DIME Analytics Stata style guide

Most academic programs that prepare students for a career in the type of work discussed in this book spend a disproportionately small amount of time teaching their students coding skills, in relation to the share of their professional time they will spend writing code their first years after graduating. Recent Masters' program graduates that have joined our team tended to have very good knowledge in the theory of our trade, but tended to require a lot of training in its practical skills. To us, it is like hiring architects that can sketch, describe, and discuss the concepts and requirements of a new building very well, but do not have the technical skill set to actually contribute to a blueprint using professional standards that can be used and understood by other professionals during construction. The reasons for this are probably a topic for another book, but in today's data-driven world, people working in quantitative economics research must be proficient programmers, and that includes more than being able to compute the correct numbers.

This appendix first has a short section with instructions on how to access and use the code shared in this book. The second section contains a the current DIME Analytics style guide for Stata code. Widely accepted and used style guides are common in most programming languages, and we think that using such a style guide greatly improves the quality of research projects coded in Stata. We hope that this guide can help to increase the emphasis in the Stata community on using, improving, sharing and standardizing code style. Style guides are the most important tool in how you, like an architect, draw a blueprint that can be understood and used by everyone in your trade.

Using the code examples in this book

You can access the raw code used in examples in this book in many ways. We use GitHub to version control everything in this book, the code included. To see the code on GitHub, go to: `https://github.com/worldbank/d4di/tree/master/code`. If you are familiar with GitHub you can fork the repository and clone your fork. We only use Stata's built-in datasets in our code examples, so you do not need to download any data from anywhere. If you have Stata installed on your computer, then you will have the data files used in the code.

A less technical way to access the code is to click the individual file in the URL above, then click the button that says **Raw**. You will then get to a page that looks like the one at: `https://raw.githubusercontent.com/worldbank/d4di/master/code/code.do`. There, you can copy the code from your browser window to your do-file editor with the formatting intact. This method is only practical

for a single file at the time. If you want to download all code used in this book, you can do that at: https://github.com/worldbank/d4di/archive/master.zip. That link offers a .zip file download with all the content used in writing this book, including the LaTeX code used for the book itself. After extracting the .zip-file you will find all the code in a folder called /code/.

Understanding Stata code

Regardless if you are new to Stata or have used it for decades, you will always run into commands that you have not seen before or do not remember what they do. Every time that happens, you should always look that command up in the helpfile. For some reason, we often encounter the conception that the helpfiles are only for beginners. We could not disagree with that conception more, as the only way to get better at Stata is to constantly read helpfiles. So if there is a command that you do not understand in any of our code examples, for example `isid`, then write `help isid`, and the helpfile for the command `isid` will open.

We cannot emphasize enough how important we think it is that you get into the habit of reading helpfiles.

Sometimes, you will encounter code employing user-written commands, and you will not be able to read those helpfiles until you have installed the commands. Two examples of these in our code are `reandtreat` or `ieboilstart`. The most common place to distribute user-written commands for Stata is the Boston College Statistical Software Components (SSC) archive. In our code examples, we only use either Stata's built-in commands or commands available from the SSC archive. So, if your installation of Stata does not recognize a command in our code, for example `randtreat`, then type `ssc install randtreat` in Stata.

Some commands on SSC are distributed in packages, for example `ieboilstart`, meaning that you will not be able to install it using `ssc install ieboilstart`. If you do, Stata will suggest that you instead use `findit ieboilstart` which will search SSC (among other places) and see if there is a package that has a command called `ieboilstart`. Stata will find `ieboilstart` in the package `ietoolkit`, so then you will type `ssc install ietoolkit` instead in Stata.

We understand that this can be confusing the first time you work with this, but this is the best way to set up your Stata installation to benefit from other people's work that they have made publicly available, and once used to installing commands like this it will not be confusing at all. All code with user-written commands, furthermore, is best written when it installs such commands at

the beginning of the master do-file, so that the user does not have to search for packages manually.

Why we use a Stata style guide

Programming languages used in computer science always have style guides associated with them. Sometimes they are official guides that are universally agreed upon, such as PEP8 for Python.[276] More commonly, there are well-recognized but non-official style guides like the JavaScript Standard Style[277] for JavaScript or Hadley Wickham's[278] style guide for R.

[276] https://www.python.org/dev/peps/pep-0008/

[277] https://standardjs.com/#the-rules

[278] http://adv-r.had.co.nz/Style.html

Aesthetics is an important part of style guides, but not the main point. The existence of style guides improves the quality of the code in that language produced by all programmers in the community. It is through a style guide that unexperienced programmers can learn from more experienced programmers how certain coding practices are more or less error-prone. Broadly-accepted style guides make it easier to borrow solutions from each other and from examples online without causing bugs that might only be found too late. Similarly, globally standardized style guides make it easier to solve each others' problems and to collaborate or move from project to project, and from team to team.

There is room for personal preference in style guides, but style guides are first and foremost about quality and standardization – especially when collaborating on code. We believe that a commonly used Stata style guide would improve the quality of all code written in Stata, which is why we have begun the one included here. You do not necessarily need to follow our style guide precisely. We encourage you to write your own style guide if you disagree with us. The best style guide woud be the one adopted the most widely. What is most important is that you adopt a style guide and follow it consistently across your projects.

The DIME Analytics Stata style guide

While this section is called a *Stata* Style Guide, many of these practices are agnostic to which programming language you are using: best practices often relate to concepts that are common across many languages. If you are coding in a different language, then you might still use many of the guidelines listed in this section, but you should use your judgment when doing so. All style rules introduced in this section are the way we suggest to code, but the most important thing is that the way you style your code is *consistent*. This guide allows our team to have a consistent code style.

Commenting code

Comments do not change the output of code, but without them, your code will not be accessible to your colleagues. It will also take you a much longer time to edit code you wrote in the past if you did not comment it well. So, comment a lot: do not only write *what* your code is doing but also *why* you wrote it like that.

There are three types of comments in Stata and they have different purposes:

1. `/* */` indicates narrative, multi-line comments at the beginning of files or sections.
2. `*` indicates a change in task or a code sub-section and should be multi-line only if necessary.
3. `//` is used for inline clarification a single line of code.

stata-comments.do

```
/*
    This is a do-file with examples of comments in Stata. This
    type of comment is used to document all of the do-file or a large
    section of it
*/

* Standardize settings (This comment is used to document a task
* covering at maximum a few lines of code)
    ieboilstart, version(13.1)
    `r(version)'

* Open the dataset
    sysuse auto.dta // Built in dataset (This comment is used to document a single line)
```

Abbreviating commands

Stata commands can often be abbreviated in the code. In the helpfiles you can tell if a command can be abbreviated, indicated by the

part of the name that is underlined in the syntax section at the top. Only built-in commands can be abbreviated; user-written commands can not. Although Stata allows some commands to be abbreviated to one or two characters, this can be confusing – two-letter abbreviations can rarely be "pronounced" in an obvious way that connects them to the functionality of the full command. Therefore, command abbreviations in code should not be shorter than three characters, with the exception of `tw` for `twoway` and `di` for `display`, and abbreviations should only be used when widely accepted abbreviation exists. We do not abbreviate `local`, `global`, `save`, `merge`, `append`, or `sort`. Here is our non-exhaustive list of widely accepted abbreviations of common Stata commands.

Abbreviation	Command
`tw`	`twoway`
`di`	`display`
`gen`	`generate`
`mat`	`matrix`
`reg`	`regress`
`lab`	`label`
`sum`	`summarize`
`tab`	`tabulate`
`bys`	`bysort`
`qui`	`quietly`
`cap`	`capture`
`forv`	`forvalues`
`prog`	`program`

Writing loops

In Stata examples and other code languages, it is common that the name of the local generated by `foreach` or `forvalues` is named something as simple as `i` or `j`. In Stata, however, loops generally index a real object, and looping commands should name that index descriptively. One-letter indices are acceptable only for general examples; for looping through **iterations** with `i`; and for looping across matrices with `i`, `j`. Other typical index names are `obs` or `var` when looping over observations or variables, respectively. But since Stata does not have arrays such abstract syntax should not be used in Stata code otherwise. Instead, index names should describe what the code is looping over, for example household members, crops, or medicines. This makes code much more readable, particularly in nested loops.

stata-loops.do

```
* This is BAD
    foreach i in potato cassava maize {
    }

* These are GOOD
    foreach crop in potato cassava maize {
    }

    * or

    local crops potato cassava maize
    * Loop over crops
    foreach crop of local crops {
        * Loop over plot number
        forvalues plot_num = 1/10 {
        }
    }
```

Using whitespace

In Stata, one space or many spaces does not make a difference, and this can be used to make the code much more readable. In the example below the exact same code is written twice, but in the good example whitespace is used to signal to the reader that the central object of this segment of code is the variable `employed`. Organizing the code like this makes the code much quicker to read, and small typos stand out much more, making them easier to spot.

We are all very well trained in using whitespace in software like PowerPoint and Excel: we would never present a PowerPoint presentation where the text does not align or submit an Excel table with unstructured rows and columns, and the same principles apply to coding.

stata-whitespace-columns.do

```
* This is BAD
    * Create dummy for being employed
    generate employed = 1
    replace employed = 0 if (_merge == 2)
    label variable employed "Person exists in employment data"
    label define yesno 1 "Yes" 0 "No"
    label value employed yesno

* This is GOOD
    * Create dummy for being employed
    generate       employed = 1
    replace        employed = 0 if (_merge == 2)
    label variable employed "Person exists in employment data"
    label define            yesno 1 "Yes" 0 "No"
    label value    employed yesno
```

Indentation is another type of whitespace that makes code

more readable. Any segment of code that is repeated in a loop or conditional on an if-statement should have indentation of 4 spaces relative to both the loop or conditional statement as well as the closing curly brace. Similarly, continuing lines of code should be indented under the initial command. If a segment is in a loop inside a loop, then it should be indented another 4 spaces, making it 8 spaces more indented than the main code. In some code editors this indentation can be achieved by using the tab button on your keyboard. However, the type of tab used in the Stata do-file editor does not always display the same across platforms, such as when publishing the code on GitHub. Therefore we recommend that indentation be 4 manual spaces instead of a tab.

stata-whitespace-indentation.do

```
* This is GOOD
    * Loop over crops
    foreach crop in potato cassava maize {
        * Loop over plot number
        forvalues plot_num = 1/10 {
            gen crop_`crop'_`plot_num' = "`crop'"
        }
    }

    * or
    local sampleSize = `c(N)'
    if (`sampleSize' <= 100) {
        gen use_sample = 0
    }
    else {
        gen use_sample = 1
    }

* This is BAD
    * Loop over crops
    foreach crop in potato cassava maize {
    * Loop over plot number
    forvalues plot_num = 1/10 {
    gen crop_`crop'_`plot_num' = "`crop'"
    }
    }

    * or
    local sampleSize = `c(N)'
    if (`sampleSize' <= 100) {
    gen use_sample = 0
    }
    else {
    gen use_sample = 1
    }
```

Writing conditional expressions

All conditional (true/false) expressions should be within at least one set of parentheses. The negation of logical expressions should use

bang (!) and not tilde (~).

stata-conditional-expressions1.do

```
* These examples are GOOD
    replace gender_string = "Female" if (gender == 1)
    replace gender_string = "Male"   if ((gender != 1) & !missing(gender))

* These examples are BAD
    replace gender_string = "Female" if gender == 1
    replace gender_string = "Male"   if (gender ~= 1)
```

You should also always use `if-else` statements when applicable even if you can express the same thing with two separate `if` statements. When using `if-else` statements you are communicating to anyone reading your code that the two cases are mutually exclusive in an `if-else` statement which makes your code more readable. It is also less error-prone and easier to update.

stata-conditional-expressions2.do

```
   local sampleSize = _N // Get the number of observations in dataset

* This example is GOOD
    if (`sampleSize' <= 100) {
    }
    else {
    }

* This example is BAD
    if (`sampleSize' <= 100) {
    }
    if (`sampleSize' > 100) {
    }
```

Using macros

Stata has several types of **macros** where numbers or text can be stored temporarily, but the two most common macros are **local** and **global**. Locals should always be the default type and globals should only be used when the information stored is used in a different do-file. Globals are error-prone since they are active as long as Stata is open, which creates a risk that a global from one project is incorrectly used in another, so only use globals where they are necessary. Our recommendation is that globals should only be defined in the **master do-file**. All globals should be referenced using both the the dollar sign and the curly brackets around their name; otherwise, they can cause readability issues when the endpoint of the macro name is unclear.

There are several naming conventions you can use for macros with long or multi-word names. Which one you use is not as important as whether you and your team are consistent in how you name then. You can use all lower case (`mymacro`), underscores (`my_macro`), or "camel case" (`myMacro`), as long as you are consistent.

stata-macros.do

```
* Define a local and a global using the same name convention
    local  myLocal  "A string local"
    global myGlobal "A string global"

* Reference the local and the global macros
    display "`myLocal'"
    display "${myGlobal}"

* Escape character. If backslashes are used just before a local
* or a global then two backslashes must be used
    local myFolderLocal  "Documents"
    local myFolderGlobal "Documents"

* These are BAD
    display "C:\Users\username\`myFolderLocal'"
    display "C:\Users\username\${myFolderGlobal}"

* These are GOOD
    display "C:\Users\username\\`myFolderLocal'"
    display "C:\Users\username\\${myFolderGlobal}"
```

Writing file paths

All file paths should be absolute and dynamic, should always be enclosed in double quotes, and should always use forward slashes for folder hierarchies (`/`), since Mac and Linux computers cannot read file paths with backslashes. File paths should also always include the file extension (`.dta`, `.do`, `.csv`, etc.), since to omit the extension causes ambiguity if another file with the same name is created (even if there is a default).

Absolute file paths means that all file paths must start at the root folder of the computer, for example `C:/` on a PC or `/Users/` on a Mac. This makes sure that you always get the correct file in the correct folder. **We never use `cd`.** We have seen many cases when using `cd` where a file has been overwritten in another project folder where `cd` was currently pointing to. Relative file paths are common in many other programming languages, but there they are relative to the location of the file running the code, and then there is no risk that a file is saved in a completely different folder. Stata does not provide this functionality.

Dynamic file paths use globals that are set in a central master do-file to dynamically build your file paths. This has the same function

in practice as setting `cd`, as all new users should only have to change these file path globals in one location. But dynamic absolute file paths are a better practice since if the global names are set uniquely there is no risk that files are saved in the incorrect project folder, and you can create multiple folder globals instead of just one location as with `cd`.

stata-filepaths.do

```
* Dynamic, absolute file paths

    * Dynamic (and absolute) - GOOD
    global myDocs    "C:/Users/username/Documents"
    global myProject "${myDocs}/MyProject"
    use "${myProject}/MyDataset.dta"

* Relative and absolute file paths

    * Relative - BAD
    cd "C:/Users/username/Documents/MyProject"
    use MyDataset.dta

    * Absolute but not dynamic - BAD
    use "C:/Users/username/Documents/MyProject/MyDataset.dta"
```

Placing line breaks

Long lines of code are difficult to read if you have to scroll left and right to see the full line of code. When your line of code is wider than text on a regular paper you should introduce a line break. A common line breaking length is around 80 characters, and Stata and other code editors provide a visible "guide line" to tell you when you should start a new line using `///`. This breaks the line in the code editor while telling Stata that the same line of code continues on the next row in the code editor. Using the `#delimit` command is only intended for advanced programming and is discouraged for analytical code in an article in Stata's official journal.[279] These do not need to be horizontally aligned in code unless they have comments, since indentations should reflect that the command continues to a new line. Line breaks and indentations can similarly be used to highlight the placement of the **option comma** in Stata commands.

[279] Cox, N. J. (2005). Suggestions on stata programming style. *The Stata Journal*, 5(4):560–566

stata-linebreak.do

```
* This is GOOD
    graph hbar ///
        invil if (priv == 1) ///
      , over(statename, sort(1) descending) blabel(bar, format(%9.0f)) ///
        ylab(0 "0%" 25 "25%" 50 "50%" 75 "75%" 100 "100%") ///
        ytit("Share of private primary care visits made in own village")

* This is BAD
    #delimit ;
    graph hbar
        invil if (priv == 1)
      , over(statename, sort(1) descending) blabel(bar, format(%9.0f))
        ylab(0 "0%" 25 "25%" 50 "50%" 75 "75%" 100 "100%")
        ytit("Share of private primary care visits made in own village");
    #delimit cr
```

Using boilerplate code

Boilerplate code is a type of code that always comes at the top of the code file, and its purpose is to harmonize the settings across users running the same code to the greatest degree possible. There is no way in Stata to guarantee that any two installations of Stata will always run code in exactly the same way. In the vast majority of cases it does, but not always, and boilerplate code can mitigate that risk (although not eliminate it). We have developed a command that runs many commonly used boilerplate settings that are optimized given your installation of Stata. It requires two lines of code to execute the `version` setting that avoids difference in results due to different versions of Stata.

stata-boilerplate.do

```
* This is GOOD
    ieboilstart, version(13.1)
    `r(version)'

* This is GOOD but less GOOD
    set more off
    set maxvar 10000
    version 13.1
```

Saving data

Similarly to boilerplate code, there are good practices that should be followed before saving the data set. These are sorting and ordering the data set, dropping intermediate variables that are not needed, and compressing the data set to save disk space and network bandwidth.

If there is an ID variable or a set of ID variables, then the code

should also test that they are uniqueally and fully identifying the data set.[280] ID variables are also perfect variables to sort on, and to order leftmost in the data set.

[280] https://dimewiki.worldbank.org/wiki/ID_Variable_Properties

The command `compress` makes the data set smaller in terms of memory usage without ever losing any information. It optimizes the storage types for all variables and therefore makes it smaller on your computer and faster to send over a network or the internet.

stata-before-saving.do

```
* If the data set has ID variables, create a local and test
* if they are fully and uniquely identifying the observations.
local idvars household_ID household_member year
isid `idvars'

* Sort and order on the idvars (or any other variables if there are no ID variables)
sort  `idvars'
order * , seq // Place all variables in alphanumeric order (optional but useful)
order `idvars' , first // Make sure the idvars are the leftmost vars when browsing

* Drop intermediate variables no longer needed

* Optimize disk space
compress

* Save data settings
save "${myProject}/myDataFile.dta" , replace // The folder global is set in master do-file
 use "${myProject}/myDataFile.dta" , clear // It is useful to be able to recall the data quickly
```

Bibliography

[1] Abadie, A. and Cattaneo, M. D. (2018). Econometric methods for program evaluation. *Annual Review of Economics*, 10:465–503.

[2] Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic control methods for comparative case studies: Estimating the effect of california's tobacco control program. *Journal of the American statistical Association*, 105(490):493–505.

[3] Abadie, A., Diamond, A., and Hainmueller, J. (2015). Comparative politics and the synthetic control method. *American Journal of Political Science*, 59(2):495–510.

[4] Abowd, J. M. (2018). The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867. ACM.

[Alecos Papadopoulos (https://stats.stackexchange.com/users/28746/alecos-papadopoulos)] Alecos Papadopoulos (https://stats.stackexchange.com/users/28746/alecos-papadopoulos). What is meant by the standard error of a maximum likelihood estimate? Cross Validated. https://stats.stackexchange.com/q/88491 (version: 2014-03-04).

[6] Angrist, J., Azoulay, P., Ellison, G., Hill, R., and Lu, S. F. (2017). Economic research evolves: Fields and styles. *American Economic Review*, 107(5):293–97.

[7] Angrist, J. D. and Krueger, A. B. (2001). Instrumental variables and the search for identification: From supply and demand to natural experiments. *Journal of Economic Perspectives*, 15(4):69–85.

[8] Angrist, J. D. and Pischke, J.-S. (2010). The credibility revolution in empirical economics: How better research design is taking the con out of econometrics. *Journal of Economic Perspectives*, 24(2):3–30.

[9] Athey, S. and Imbens, G. W. (2017a). The econometrics of randomized experiments. *Handbook of Economic Field Experiments*, 1:73–140.

[10] Athey, S. and Imbens, G. W. (2017b). The state of applied econometrics: Causality and policy evaluation. *Journal of Economic Perspectives*, 31(2):3–32.

[11] Banerjee, A. V. and Duflo, E. (2009). The experimental approach to development economics. *Annual Review of Economics*, 1(1):151–178.

[12] Begg, C., Cho, M., Eastwood, S., Horton, R., Moher, D., Olkin, I., Pitkin, R., Rennie, D., Schulz, K. F., Simel, D., et al. (1996). Improving the quality of reporting of randomized controlled trials: The CONSORT statement. *JAMA*, 276(8):637–639.

[13] Bound, J., Jaeger, D. A., and Baker, R. M. (1995). Problems with instrumental variables estimation when the correlation between the instruments and the endogenous explanatory variable is weak. *Journal of the American Statistical Association*, 90(430):443–450.

[14] Calonico, S., Cattaneo, M. D., Farrell, M. H., and Titiunik, R. (2019). Regression discontinuity designs using covariates. *Review of Economics and Statistics*, 101(3):442–451.

[15] Carril, A. (2017). Dealing with misfits in random treatment assignment. *The Stata Journal*, 17(3):652–667.

[16] Christensen, G. and Miguel, E. (2018). Transparency, reproducibility, and the credibility of economics research. *Journal of Economic Literature*, 56(3):920–80.

[17] Cox, N. J. (2005). Suggestions on stata programming style. *The Stata Journal*, 5(4):560–566.

[18] Dafoe, A. (2014). Science deserves better: the imperative to share complete replication files. *PS: Political Science & Politics*, 47(1):60–66.

[19] DiNardo, J. (2016). Natural experiments and quasi-natural experiments. *The New Palgrave Dictionary of Economics*, pages 1–12.

[20] Duflo, E., Glennerster, R., and Kremer, M. (2007). Using randomization in development economics research: A toolkit. *Handbook of Development Economics*, 4:3895–3962.

[21] Duvendack, M., Palmer-Jones, R., and Reed, W. R. (2017). What is meant by "replication" and why does it encounter resistance in economics? *American Economic Review*, 107(5):46–51.

[22] Flom, P. (2005). LaTeX for academics and researchers who (think they) don't need it. *The PracTEX Journal*, 4.

[23] Gobillon, L. and Magnac, T. (2016). Regional policy evaluation: Interactive fixed effects and synthetic controls. *Review of Economics and Statistics*, 98(3):535–551.

[24] Hausman, C. and Rapson, D. S. (2018). Regression discontinuity in time: Considerations for empirical applications. *Annual Review of Resource Economics*, 10:533–552.

[25] Healy, K. (2018). *Data visualization: A practical introduction*. Princeton University Press.

[26] Iacus, S. M., King, G., and Porro, G. (2012). Causal inference without balance checking: Coarsened exact matching. *Political Analysis*, 20(1):1–24.

[27] Imbens, G. W. and Lemieux, T. (2008). Regression discontinuity designs: A guide to practice. *Journal of Econometrics*, 142(2):615–635.

[28] Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Medicine*, 2(8):e124.

[29] Ioannidis, J. P., Stanley, T. D., and Doucouliagos, H. (2017). The power of bias in economics research. *The Economic Journal*.

[30] King, G. and Nielsen, R. (2019). Why propensity scores should not be used for matching. *Political Analysis*, 27(4):435–454.

[31] Lee, D. S. and Lemieux, T. (2010). Regression discontinuity designs in economics. *Journal of Economic Literature*, 48(2):281–355.

[32] Levitt, S. D. and List, J. A. (2009). Field experiments in economics: The past, the present, and the future. *European Economic Review*, 53(1):1–18.

[33] Matthews, G. J., Harel, O., et al. (2011). Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Surveys*, 5:1–29.

[34] McKenzie, D. (2012). Beyond baseline and follow-up: The case for more T in experiments. *Journal of Development Economics*, 99(2):210–221.

[35] Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., et al. (2015). Promoting an open research culture. *Science*, 348(6242):1422–1425.

[36] Olken, B. A. (2015). Promises and perils of pre-analysis plans. *Journal of Economic Perspectives*, 29(3):61–80.

[37] Orozco, V., Bontemps, C., Maigne, E., Piguet, V., Hofstetter, A., Lacroix, A., Levert, F., Rousselle, J.-M., et al. (2018). How to make a pie? reproducible research for empirical economics & econometrics. *Toulouse School of Economics Working Paper*, 933.

[38] Rogers, A. (2017). The dismal science remains dismal, say scientists. *Wired*.

[39] Schulz, K. F., Altman, D. G., and Moher, D. (2010). Consort 2010 statement: updated guidelines for reporting parallel group randomised trials. *BMC medicine*, 8(1):18.

[40] Simmons, J. P., Nelson, L. D., and Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11):1359–1366.

[41] Simonsohn, U., Nelson, L. D., and Simmons, J. P. (2014). P-curve: a key to the file-drawer. *Journal of Experimental Psychology: General*, 143(2):534.

[42] Simonsohn, U., Simmons, J. P., and Nelson, L. D. (2015). Specification curve: Descriptive and inferential statistics on all reasonable specifications. *Available at SSRN 2694998*.

[43] Stock, J. and Yogo, M. (2005). *Testing for Weak Instruments in Linear IV Regression*, pages 80–108. Cambridge University Press, New York.

[44] Stodden, V., Guo, P., and Ma, Z. (2013). Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals. *PloS one*, 8(6):e67111.

[45] Wicherts, J. M., Veldkamp, C. L., Augusteijn, H. E., Bakker, M., Van Aert, R., and Van Assen, M. A. (2016). Degrees of freedom in planning, running, analyzing, and reporting psychological studies: A checklist to avoid p-hacking. *Frontiers in Psychology*, 7:1832.

[46] Wickham, H. (2014). Tidy data. *The Journal of Statistical Software*, 59.

[47] Wilke, C. O. (2019). *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. O'Reilly Media.

[48] Young, A. (2017). Consistency without inference: Instrumental variables in practical application. *Unpublished manuscript, London: London School of Economics and Political Science. Retrieved from:* `http://personal.lse.ac.uk/YoungA`.

Index