DIME analytics

KRISTOFFER BJÄRKEFUR

LUÍZA CARDOSO DE ANDRADE

BENJAMIN DANIELS

MARIA JONES

DATA FOR DEVELOPMENT IMPACT

DIME ANALYTICS RESOURCE GUIDE

WORLD BANK GROUP

i2i DIME TRANSFORM DEVELOPMENT

UKaid from the British people

Norad

15173-Analytics_coverFINAL.indd 1 5/17/19 2:45 PM

KRISTOFFER BJÄRKEFUR
LUÍZA CARDOSO DE ANDRADE
BENJAMIN DANIELS
MARIA JONES

DATA FOR DEVELOPMENT IMPACT: THE DIME ANALYTICS RESOURCE GUIDE

DIME ANALYTICS

Copyright © 2020
Kristoffer Bjärkefur
Luíza Cardoso de Andrade
Benjamin Daniels
Maria Jones

PUBLISHED BY DIME ANALYTICS

https://worldbank.github.com/d4di

Released under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

https://creativecommons.org/licenses/by/4.0

First printing, February 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

11 Introduction: Data for development impact

17 Chapter 1: Handling data ethically

27 Chapter 2: Collaborating on code and data

43 Chapter 3: Evaluating impact through research design

55 Chapter 4: Sampling, randomization, and power

69 Chapter 5: Acquiring development data

87 Chapter 6: Analyzing research data

103 Chapter 7: Publishing collaborative research

115 Bringing it all together

117 Appendix: The DIME Analytics Stata Style Guide

131 Bibliography

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 teach all users of development data how to handle data effectively, efficiently, and ethically. An empirical revolution has changed the face of research economics rapidly over the last decade. Today, especially in the development subfield, working with raw data – whether collected through surveys or acquired from "big" data sources like sensors, satellites, or call data records – is a key skill for researchers and their staff. At the same time, the scope and scale of empirical research projects is expanding: more people are working on the same data over longer timeframes. As the ambition of development researchers grows, so too has the complexity of the data on which they rely to make policy-relevant research conclusions. Yet there are few guides to the conventions, standards, and best practices that are fast becoming a necessity for empirical research. This book aims to fill that gap.

This book is targeted to everyone who interacts with development data: graduate students, research assistants, policymakers, and empirical researchers. It covers data workflows at all stages of the research process, from design to data acquisition and analysis. Its content is not sector-specific; it will not teach you econometrics, or how to design an impact evaluation. There are many excellent existing resources on those topics. Instead, this book will teach you how to think about all aspects of your research from a data perspective, how to structure research projects to maximize data quality, and how to institute transparent and reproducible workflows. The central premise of this book is that data work is 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 research projects. It aims to be a highly practical resource: we provide code snippets, links to checklists and other practical tools, and references to primary resources that allow the reader to immediately put recommended processes into practice.

Doing credible research at scale

The team responsible for this book is known as **DIME Analytics**.[1] The DIME Analytics team is part of the **Development Impact Evaluation (DIME)** Department[2] within the World Bank's **Development Economics (DEC) Vice Presidency**.[3]

DIME generates high-quality and operationally relevant data and research to transform development policy, help reduce extreme poverty, and secure shared prosperity. It develops customized data and evidence ecosystems to produce actionable information and recommend specific policy pathways to maximize impact. DIME conducts research in 60 countries with 200 agencies, leveraging a

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

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

[3] https://www.worldbank.org/en/about/unit/unit-dec

US$180 million research budget to shape the design and implementation of US$18 billion in development finance. DIME also provides advisory services to 30 multilateral and bilateral development agencies. Finally, DIME invests in public goods (such as this book) to improve the quality and reproducibility of development research around the world.

DIME Analytics was created to take advantage of the concentration and scale of research at DIME to develop and test solutions, to ensure high quality data collection and research across the DIME portfolio, and to make training and tools publicly available to the larger community of development researchers. *Data for Development Impact* compiles the ideas, best practices and software tools Analytics has developed while supporting DIME's global impact evaluation portfolio.

The **DIME Wiki** is one of our flagship products, a free online collection of our resources and best practices.[4] This book complements the DIME Wiki by providing a structured narrative of the data workflow for a typical research project. We will not give a lot of highly specific details in this text, but we will point you to where they can be found.[5] Each chapter focuses on one task, providing a primarily narrative account of: what you will be doing; where in the workflow this task falls; when it should be done; and how to implement it according to best practices.

[4] https://dimewiki.worldbank.org

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

We will use broad terminology throughout this book to refer to research team members: **principal investigators (PIs)** who are responsible for the overall design and stewardship of the study; **field coordinators (FCs)** who are responsible for the implementation of the study on the ground; and **research assistants (RAs)** who are responsible for handling data processing and analytical tasks.

Adopting reproducible tools

We assume througout all of this book that you are going to do nearly all of your data work though code. It may be possible to perform all relevant tasks through the user interface in some statistical software, or even through less field-specific software such as Excel. However, we strongly advise against it. The reason for that are the transparency, reproducibility and credibility principles discussed in Chapter 1. Writing code creates a record of every task you performed. It also prevents direct interaction with the data files that could lead to non-reproducible processes. Think of the code as a recipe to create your results: other people can follow it, reproduce it, and even disagree with your the amount of spices you added (or some of your coding decisions). Many development researchers come from

economics and statistics backgrounds and often understand code to be a means to an end rather than an output itself. We believe that this must change somewhat: in particular, we think that development practitioners must begin to think about their code and programming workflows just as methodologically as they think about their research workflows.

Most tools have a learning and adaptation process, meaning you will become most comfortable with each tool only by using it in real-world work. To support your process of learning reproducible tools and workflows, will reference free and open-source tools wherever possible, and point to more detailed instructions when relevant. Stata, as a proprietary software, is the notable exception here due to its current popularity in development economics.[6] This book also includes the DIME Analytics 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 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.

[6] https://aeadataeditor.github.io/presentation-20191211/#9

Writing reproducible code in a collaborative environment

Throughout the book, we refer to the importance of good coding practices. These are the foundation of reproducible and credible data work, and a core part of the new data science of development research. Code today is no longer a means to an end (such as a research paper), rather it is part of the output itself: a means for communicating how something was done, in a world where the credibility and transparency of data cleaning and analysis is increasingly important. As this is fundamental to the remainder of the book's content, we provide here a brief introduction to **"good" code** and **process standardization**.

"Good" code has two elements: (1) it is correct, i.e. it doesn't produce any errors, and (2) it is useful and comprehensible to someone who hasn't seen it before (or even yourself a few weeks, months or years later). Many researchers have been trained to code correctly. However, when your code runs on your computer and you get the correct results, you are only half-done writing *good* code. Good code is easy to read and replicate, making it easier to spot mistakes. Good code reduces sampling, randomization, and cleaning errors. Good code can easily be reviewed by others before it's published and replicated afterwards.

Process standardization means that there is little ambiguity about how something ought to be done, and therefore the tools to do it

can be set in advance. Standard processes for code help other people to ready your code.[7] Code should be well-documented, contain extensive comments, and 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.[8]

[7] https://dimewiki.worldbank.org/Stata_Coding_Practices

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

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 could 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.

As you gain experience in coding and get more confident with the way you implement these suggestions, you will feel more empowered to apply critical thinking to the way you handle data. For example, you will be able to predict which section of your script are more likely to create errors. This may happen intuitively, but you will improve much faster as a coder if you do it purposefully. Ask yourself, as you write code and explore results: Do I believe this number? What can go wrong in my code? How will missing values be treated in this command? What would happen if more observations would be added to the dataset? Can my code be made more efficient or easier to understand?

Code examples

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. All code guidance is software-agnostic, but code examples are provided in Stata. In the book, code examples 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 ensure that each code block runs independently, is well-formatted, and uses built-in functions as much as possible. We will point to user-written functions when they provide important tools. In particular, we point to two suites of Stata commands developed by DIME Analytics, `ietoolkit`[9] and `iefieldkit`,[10] which standardize our core data collection, management, and analysis workflows. We will comment the code generously (as you should), but you should reference Stata help-files by writing `help [command]` whenever you do not understand the command that is being used. We hope that these snippets will provide a foundation for your code style. Providing some standardization to Stata code style is also a goal of this team; we provide our guidance on this in the Stata Style Guide in the Appendix.

[9] https://dimewiki.worldbank.org/ietoolkit

[10] https://dimewiki.worldbank.org/iefieldkit

Outline of this book

This book covers each stage of an empirical research project, from design to publication. We start with ethical principles to guide empirical research, focusing on research transparency and the right to privacy. In Chapter 1, we outline a set of practices that help to ensure research participants are appropriately protected and research consumers can be confident in the conclusions reached. Chapter 2 will teach you to structure your data work to be efficient, collaborative and reproducible. It discusses the importance of planning data work at the outset of the research project – long before any data is acquired – and provides suggestions for collaborative workflows and tools. In Chapter 3, we turn to research design, focusing specifically on how to measure treatment effects and structure data for common experimental and quasi-experimental research methods. We provide an overview of research designs frequently used for causal inference, and consider implications for data structure. Chapter 4 concerns sampling and randomization:

how to implement both simple and complex designs reproducibly, and how to use power calculations and randomization inference to critically and quantitatively assess sampling and randomization to make optimal choices when planning studies.

Chapter 5 covers data acquisition. We start with the legal and institutional frameworks for data ownership and licensing, dive in depth on collecting high-quality survey data, and finally discuss secure data handling during transfer, sharing, and storage. Chapter 6 teaches reproducible and transparent workflows for data processing and analysis, and provides guidance on de-identification of personally-identified data, focusing on how to organize data work so that it is easy to code the desired analysis. In Chapter 7, we turn to publication. You will learn how to effectively collaborate on technical writing, how and why to publish data, and guidelines for preparing functional and informative replication packages.

While adopting the workflows and mindsets described in this book requires an up-front cost, it will save you (and your collaborators) a lot of time and hassle very quickly. In part this is because you will learn how to implement essential practices directly; in part because you will find tools for the more advanced practices; and most importantly because you will acquire the mindset of doing research with a high-quality data focus. We hope you will find this book helpful for accomplishing all of the above, and that mastery of data helps you make an impact. We hope that by the end of the book, you will have learned how to handle data more efficiently, effectively and ethically at all stages of the research process.

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 too does the (rightful) scrutiny under which methods and results are placed. It is useful to think of research as a public service, one that requires you to be accountable to both research participants and research consumers. On the research participant side, it is essential to respect individual privacy and ensure data security. Researchers look deeply into real people's personal lives, financial conditions, and other sensitive subjects. Respecting the respondents' right to privacy, by intelligently assessing and proactively averting risks they might face, is a core tenet of research ethics. On the consumer side, it is important to protect confidence in development research by following modern practices for transparency and reproducibility.

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. Neither privacy nor transparency is an all-or-nothing objective: the most important thing is to report the transparency and privacy measures you have taken and always strive to do the best that you are capable of with current technology. In this chapter, we outline a set of practices that help to ensure research participants are appropriately protected and research consumers can be confident in the conclusions reached. Later chapters will provide more hands-on guides to implementing those practices.

Protecting confidence in development research

The empirical revolution in development research[1] has therefore led to increased public scrutiny of the reliability of research.[2] Three major components make up this scrutiny: **reproducibility**[3], **transparency**,[4] and **credibility**.[5] 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. As a result, it is often the case that the data 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.

[1] 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

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

[3] 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

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

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

Reproducible and transparent methods are key to maintaining credibility and avoiding serious errors. This is particularly true for research that relies on original or novel data sources, from innovative big data sources to surveys. The field is slowly moving in the direction of requiring greater transparency. Major publishers and funders, most notably the American Economic Association, have taken steps to require that code and data are accurately reported, cited, and preserved as outputs in themselves.[6]

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

Research reproducibility

Can another researcher reuse the same code on the same data and get the exact same results as in your published paper?[7] This is a standard known as **computational reproducibility**, and it is an increasingly common requirement for publication.[8]) It is best practice to verify computational reproducibility before submitting a paper before publication. This should be done by someone who is not on your research team, on a different computer, using exactly the package of code and data files you plan to submit with your paper. Code that is well-organized into a master script, and written to be easily run by others, makes this task simpler. The next chapter discusses organization of data work in detail.

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

[8] https://www.nap.edu/resource/25303/R&R.pdf

For research to be reproducible, all code files for 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.[9] 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.[10]

[9] 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

Making your research reproducible is also a public good.[11] It 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

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

[11] https://dimewiki.worldbank.org/Reproducible_Research

corresponding dataset should be openly accessible unless for legal or ethical reasons it cannot be.[12]

[12] https://dimewiki.worldbank.org/Publishing_Data

Research transparency

Transparent research will expose not only the code, but all research processes involved in developing the analytical approach.[13] This means that readers are 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[14] 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 over the complete course of a project.

[13] https://www.princeton.edu/~mjs3/open_and_reproducible_opr_2017.pdf

[14] https://dimewiki.worldbank.org/Data_Documentation

Tools like **pre-registration**[15], **pre-analysis plans**[16], and **registered reports**[17] can help with this process where they are available. By pre-specifying a large portion of the research design,[18] 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",[19] 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.[20] 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.

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

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

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

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

[19] 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

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

Documenting a project in detail greatly increases transparency. Many disciplines have a tradition of keeping a "lab notebook", and adapting and expanding this process to create a lab-style workflow in the development field is a critical step towards more transparent practices. 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.

There are various software solutions for building documentation over time. The **Open Science Framework**[21] provides one such solution, with integrated file storage, version histories, and collaborative wiki pages. **GitHub**[22] provides a transparent documentation system[23], in addition to version histories and wiki pages. Such services offer 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. Services that log your research process can show things like modifications made in response to referee comments, by having tagged version histories at each major revision. They also allow you to use issue trackers to document the research paths and questions you may have tried to answer as a resource to others who have similar questions. Each project has specific requirements for data, code, and documentation management, and the exact transparency tools to use will depend on the team's needs, but they 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. Email, however, is *not* a note-taking service, because communications are rarely well-ordered, can be easily deleted, and are not available for future team members.

[21] https://osf.io

[22] https://github.com

[23] https://dimewiki.worldbank.org/Getting_started_with_GitHub

Research credibility

The credibility of research is traditionally a function of design choices.[24] 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? Pre-analysis plans can be used to assuage these concerns for experimental evaluations by fully specifying some set of analysis intended to be conducted. Regardless of whether or not a formal pre-analysis plan is utilized, all experimental and observational studies should be pre-registered simply to create a record of the fact that the study was undertaken.[25] This is increasingly required by publishers and can be done very quickly using the **AEA** database,[26] the **3ie** database,[27] the **eGAP** database,[28] or the **OSF** registry,[29] as appropriate.

Common research standards from journals and funders feature both ex ante (or "regulation") and ex post (or "verification") policies.[30] Ex ante policies require that authors bear the burden of ensuring they provide some set of materials before publication and their quality

[24] 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

[25] https://datacolada.org/12

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

[27] https://ridie.3ieimpact.org

[28] https://egap.org/content/registration

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

[30] 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

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.[31]

[31] 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

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.

Ensuring privacy and security in research data

Anytime you are working with original data in a development research project, you are almost certainly handling data that include **personally-identifying information (PII)**.[32] PII data contains information that can, without any transformation, be used to identify individual people, households, villages, or firms that were part of 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 data privacy principles not only for the respondent but also the PII data of their household members or other individuals 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 general. There is no one-size-fits-all solution to determine what is

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

PII, and you will have to use careful judgment in each case to decide which pieces of information fall into this category.[33]

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".[34] If you interact with European institutions or persons, you will also become familiar with the General Data Protection Regulation (GDPR),[35] a set of regulations governing **data ownership** and privacy standards.[36]

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. Even if your research does not involve PII, it is a prerogative of the data owner to determine who may have access to it. Therefore, if you are using data that was provided to you by a partner, they have the right to request that you hold it to uphold the same data security safeguards as you would to PII. For the purposes of this book, we will call all any data that may not be freely accessed for these or other reasons **confidential data**. Given 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 for any action that knowingly or recklessly ignores these considerations.

[33] https://sdcpractice.readthedocs.io

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

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

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

Obtaining ethical approval and consent

For almost all data collection and research activities that involve human subjects or PII data, you will be required to complete some form of **Institutional Review Board (IRB)** process.[37] Most commonly this consists of a formal application for approval of a specific protocol for consent, data collection, and data handling.[38] Which IRB 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 local institution subject to the laws of the country where data originates.

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

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

[38] https://dimewiki.worldbank.org/IRB_Approval

responsible to EU law) view all personal data as 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. All survey instruments must include a module in which the sampled respondent grants informed consent to participate. Research participants must be informed of the purpose of the research, what their participation will entail in terms of duration and any procedures, any foreseeable benefits or risks, and how their identity will be protected.[39] 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.

[39] https://www.icpsr.umich.edu/icpsrweb/content/datamanagement/confidentiality/conf-language.html

IRB approval should be obtained well before any data is acquired. 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 possible 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 acquired 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.[40] There are several precautions needed to ensure that your data is safe. 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. However, password-protection alone is not sufficient, because if the underlying data is obtained through a leak the information itself remains usable. Datasets that include confidential information *must* therefore be **encrypted**[41] during data collection, storage, and transfer.

[40] https://dimewiki.worldbank.org/Data_Security

Most modern data collection software has features that, if enabled,

[41] **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/Encryption

make secure transmission straightforward.[42] Many also have features that ensure data is encrypted when stored on their servers, although this usually needs to be actively enabled and administered.[43] When files are properly encrypted, the information they contain will be completely unreadable and unusable even if they were to be intercepted my a malicious "intruder" or accidentally made public. When the proper data security precautions are taken, no one who is not listed on the IRB may have access to the decryption key. This means that is it usually not enough to rely service providers' on-the-fly encryption as they need to keep a copy of the decryption key to make it automatic. When confidential data is stored on a local computer it must always remain encrypted, and confidential data may never be sent unencrypted over email, WhatsApp, or other chat services.

[42] https://dimewiki.worldbank.org/Encryption#Encryption_in_Transit

[43] https://dimewiki.worldbank.org/Encryption#Encryption_at_Rest

The easiest way to reduce the risk of leaking confidential information is to use it as rarely as possible. It is often very simple to conduct planning and analytical work using a subset of the data that does not include this type of information. We encourage this approach, because it is easy. However, when confidential data is absolutely necessary to a task, such as implementing an intervention or submitting survey data, you must actively protect that information 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 all confidential information, especially when a worst-case scenario could potentially lead to re-identifying subjects. Extremely confidential 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. What data security protocols you employ will depend on project needs and data sources, but agreeing on a protocol from the start of a project will make your life easier. 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 data

Most of the field research done in development involves human subjects.[44] As a researcher, you are asking people to trust you with

[44] https://dimewiki.worldbank.org/Human_Subjects_Approval

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.[45] Your donor or employer will most likely require you to hold a certification from a source such as Protecting Human Research Participants[46] or the CITI Program.[47]

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 never be more than one copy of the raw identified dataset in the project folder, and it must always be encrypted. Even within the research team, access to PII data should be limited to team members who require it for specific analysis (most analysis will not depend on PII). 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, that is, remove direct identifiers of the individuals in the dataset.[48] 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. For this reason, we recommend de-identification in two stages. The **initial de-identification** process strips the data of direct identifiers as early in the process as possible, to create a working de-identified dataset that can be shared *within the research team* without the need for encryption. This simplifies workflows. The **final de-identification** process involves making a decision about the trade-off between risk of disclosure and utility of the data before publicly releasing a dataset.[49] We will provide more detail about the process and tools available for initial and final de-identification in Chapters 6 and 7, respectively.

[45] https://dimewiki.worldbank.org/Research_Ethics

[46] https://phrptraining.com

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

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

[49] https://sdcpractice.readthedocs.io/en/latest/SDC_intro.html#need-for-sdc

Chapter 2: Collaborating on code and data

Preparation for collaborative data work begins long before you acquire any data, and involves planning both the software tools you will use 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 structure your workflow in advance. This means knowing which datasets and outputs you need at the end of the process, how they will stay organized, what types of data you'll acquire, and whether the data will require special handling due to size or privacy considerations. Identifying these details will help you map out the data needs for your project, and give you a sense for how information resources should be organized. It's okay to update this data map once the project is underway. The point is that everyone knows – at any given time – what the plan is.

To do data work effectively in a team environment, 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 plan ahead. Seemingly small decisions such as sharing services, folder structures, and filenames can be extremely painful to alter down the line in any project. Similarly, make sure to set up a self-documenting discussion platform and process for version control; this makes working together on outputs much easier from the very first discussion. This chapter will guide you on preparing a collaborative work environment, and structuring your data work to be well-organized and clearly documented. It outline hows to set up your working environment and prepare to collaborate on technical tasks with others, as well as how to document tasks and decisions. It then discusses how to keep code, data, and outputs organized so that others will be able to locate and work with materials easily.

Preparing a collaborative work environment

This section provides a brief introduction to core concepts and tools that can help you to organize your data work in an efficient, collaborative and reproducible manner. 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; those add up to substantial gains over the course of multiple years and 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. Adaptation is good, of course. But it is important to recognize that there are a number of tasks that exist for

every project, and it is more efficient to agree on the corresponding workflows 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 section 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 available on many modern operating systems; you should determine whether your computer implements this or whether you need to ensure encryption at the individual file level. 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.[1] 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.

[1] 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 file paths 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 several 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[2] and GitHub[3]. 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.

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

[3] **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.[4]

[4] https://dimewiki.worldbank.org/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".[5] 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 chronological. 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.

[5] 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.[6])

[6] https://dimewiki.worldbank.org/ieboilstart

Next, think about how and where you write and execute code. This book is intended to be agnostic to the size or origin of your data, but we are going to broadly assume that you are using desktop-sized 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.[7] For Stata,

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

the built-in do-file editor is the most widely adopted code editor, but **Atom**[8] and **Sublime**[9] 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.

[8] https://atom.io

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

Organizing code and data

We assume you are going to do nearly all of your analytical work through code. Though it is possible to use some statistical software through the user interface without writing any code, we strongly advise against it. Writing code creates a record of every task you performed. It also prevents direct interaction with the data files that could lead to non-reproducible steps. You may do some exploratory tasks by point-and-click or typing directly into the console, but anything that is included in a research output must be coded up in an organized fashion so that you can release the exact code recipe that goes along with your final results. Still, organizing code and data into 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`[10] as a part of our `ietoolkit` package. This Stata command sets up a pre-standardized folder structure for what we call the `DataWork` folder.[11] 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.

[10] https://dimewiki.worldbank.org/iefolder

[11] https://dimewiki.worldbank.org/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 or survey round. 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 has some flexibility for the addition of folders for other types of data sources, although this is less well developed as the needs for larger datasets tend to be very specific. 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 written in a plaintext language called **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 is preferable to create the `DataWork` folder separately from the project management materials (such as 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-code-compatiable 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 code folder than in the project folder.

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 code-compatiable 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.

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: "code-compatible" files, which are those that are accessed by code processes, and "non-code-compatible" files, which will not be accessed by code processes. The former takes precedent: an Excel file is a code-compatible 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.[12]

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

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. At the end of this section, we include a template for a master script in Stata, to provide a concrete example of the required elements and structure. Throughout this section, we refer to lines of the example do-file to give concrete examples of the required code elements, organization and structure.

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**[13] that details the functionality of the entire script; refer to lines 5-10 in the example do-file. 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. You should always track the inputs and outputs of the script, as well as the uniquely identifying variable; refer to lines 49-51 in the example do-file. When you are trying to track down which code creates which dataset, this will be very helpful. While there are other ways to document decisions related to creating code, the information that is relevant to understand the code should always be written in the code file.

[13] **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; refer to line 35 in the example do-file. 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 in) 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 easy-to-find location and naming them in a meaningful way. 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: Prepare folder paths and define programs`, 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; refer to line 24 of the example do-file. You should also add an index in the code header by copying and pasting section titles; refer to lines 8-10 in the example do-file. You can then add and navigate through them using the `find` functionality. 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 common practice of limiting code lines to 80 characters: it seems to be "enough but not too much" for most purposes.

Working with a master script

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.

Try to create the habit of running your code from the master script. Creating "section switches" using macros or objects to run only the codes related to a certain task should always be preferred to manually open different scripts to run them in a certain order (see Part 1 of `stata-master-dofile.do` for an example of how to do this). Furthermore, running all scripts related to a particular task through the master whenever one of them is edited helps you identify unintended consequences of the changes you made. Say, for example, that you changed the name of a variable created in one script. This may break another script that refers to this variable. But unless you run both of them when the change is made, it may take time for that to happen, and when it does, it may take time for you to understand what's causing an error. The same applies to changes in datasets and results.

To link code, data and outputs, the master script reflects the structure of the `DataWork` folder in code through globals (in Stata) or string scalars (in R); refer to lines 35-40 of the example do-file. 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; refer to lines 27-32 of the example do-file. The code in `stata-master-dofile.do` shows how folder structure is reflected in a master do-file. Because writing and maintaining a master script can be challenging as a project grows,

an important feature of the `iefolder` is to write master do-files and add to them whenever new subfolders are created in the `DataWork` folder.[14]

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

In order to maintain well-documented and organized code, 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 is 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.

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:  Install user-written packages and harmonize settings
*******************************************************************************/

    local user_commands ietoolkit iefieldkit //Add required user-written commands
    foreach command of local user_commands {
        cap which `command'
        if _rc == 111 ssc install `command'
    }

        *Harmonize settings accross users as much as possible
    ieboilstart, v(13.1)
    `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   "M:/DataWork/EncryptedData"
    }


    * 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:    ${bl_dt}/Raw Deidentified/D4DI_baseline_raw_deidentified.dta
 IDS VAR:    hhid
------------------------------------------------------------------------------ */
    if (0) { //Change the 0 to 1 to run the baseline de-identification dofile
        do "${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
              ${bl_doc}/Codebook baseline.xlsx
  IDS VAR:    hhid
------------------------------------------------------------------------------ */
    if (0) { //Change the 0 to 1 to run the baseline cleaning dofile
        do "${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
------------------------------------------------------------------------------ */
```

Managing outputs

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 `f1-factor-analysis.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 datasets 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.[15] LaTeX is a document preparation system that can create both text documents and presentations. 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[16].

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[17] is the most widely adopted solution in R. There are also different options for Markdown in Stata, such as `markstat`,[18] Stata 15 dynamic documents,[19] `webdoc`,[20] and `texdoc`.[21]

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.

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

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

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

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

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

[20] http://repec.sowi.unibe.ch/stata/webdoc

[21] http://repec.sowi.unibe.ch/stata/texdoc

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 research design to do effective data work, but it is essential that you understand the design of the study you are working on, and how it affects the data work. Without going into too much technical detail, as there are many excellent resources on impact evaluation design, this chapter presents a brief overview of the most common causal inference methods, focusing on implications for data structure and analysis. 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 specific code tools designed for each method (the list, of course, is not exhaustive).

Thinking through research 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 produce meaningful analytics throughout your projects. 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.

This chapter first covers causal inference methods. Next it discusses how to measure treatment effects and structure data for specific methods, including cross-sectional randomized control trials, difference-in-difference designs, regression discontinuity, instrumental variables, matching, and synthetic controls.

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.[1] 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

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

precise estimate of that treatment effect. Under this condition, we can proceed to draw evidence from the limited samples we have 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. 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[2] 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 causal identification in different contexts.

[2] 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, by establishing a **counterfactual**[3] for the treatment group against 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.[4] *Causal Inference* and *Causal Inference: The Mixtape* provides more detailed mathematical approaches to the tools.[5] *Mostly Harmless Econometrics* and *Mastering Metrics* are excellent resources on the statistical principles behind all econometric approaches.[6]

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.[7] Instead, we typically make inferences from samples. **Causal inference** methods are those in which we are able to estimate the average

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

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

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

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

[7] https://www.stat.columbia.edu/~cook/qr33.pdf

treatment effect without observing individual-level effects, but through some comparison of averages with a **control** group. Every 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.[8]

[8] https://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. Second, 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,[9] often in the form of government programs, NGO projects, new regulations, information campaigns, and many more types of interventions.[10] The classic experimental causal inference method is the **randomized control trial (RCT)**.[11] In randomized control trials, the treatment group is randomized – that is, from an eligible population, a random group of units are given the treatment. Another way to think about

[9] https://dimewiki.worldbank.org/Experimental_Methods

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

[11] https://dimewiki.worldbank.org/Randomized_Control_Trials

these designs is how they establish the control group: 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 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.[12] 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.[13]

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

[13] 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, take-up 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% take-up or efficacy doubles the required sample, and 50% quadruples it.[14] Such effects are also very hard to correct ex post, since they require strong assumptions about the randomness or non-randomness of take-up. 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 may overshadow the effort put into the econometric design itself.

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

Quasi-experimental research designs,[15] 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.[16] 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 or will occur. Therefore, these methods often use either secondary data, or they use primary data in a cross-sectional retrospective method, including administrative data or other new classes of routinely-collected information.

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

[16] 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 observes 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 to track individuals across time. 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 experimental, the treatment and control groups are randomly constructed from the population that is eligible to receive each treatment. 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.

Cross-sectional designs can also exploit variation in non-experimental data to argue that observed correlations do in fact represent causal effects. This can be true unconditionally – which is to say that something random, such as winning the lottery, is a true random process and can tell you about the effect of getting a large amount of money.[17] It can also be true conditionally – which is to say that once the characteristics that would affect both the likelihood of exposure to a treatment and the outcome of interest are controlled for, the process is as good as random: like arguing that once risk preferences are taken into account, exposure to an earthquake is unpredictable and post-event differences are causally related to the event itself.[18]

For cross-sectional designs, what needs to be carefully maintained in data is the treatment randomization process itself (whether experimental or not), as well as detailed information about differences

[17] Imbens, G. W., Rubin, D. B., and Sacerdote, B. I. (2001). Estimating the effect of unearned income on labor earnings, savings, and consumption: Evidence from a survey of lottery players. *American economic review*, 91(4):778–794

[18] Callen, M. (2015). Catastrophes and time preference: Evidence from the indian ocean earthquake. *Journal of Economic Behavior & Organization*, 118:199–214

in data quality and attrition across groups.[19] 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).[20] **Randomization inference** can be used to estimate the underlying variability in the randomization process (more on this in the next chapter). **Balance checks**[21] 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.[22]

Analysis is typically straightforward *once you have a strong understanding of the randomization*. A typical analysis will include a description 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,[23] including to analyze balance[24] and to visualize treatment effects.[25] Extensive tools and methods for analyzing selective non-response are available.[26]

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**[27] 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.[28] 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").[29] The regression model includes a control variable for treatment

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

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

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

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

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

[24] https://dimewiki.worldbank.org/iebaltab

[25] https://dimewiki.worldbank.org/iegraph

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

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

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

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

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 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.[30] Experimental approaches satisfy this requirement in expectation, but a given randomization should still be checked for pre-trends as an extension of balance checking.[31]

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).[32] This allows each unit's baseline outcome (the outcome before the intervention) to be used as an additional control for its endline outcome, which can provide large increases in power and robustness.[33] When tracking individuals over time for this purpose, maintaining sampling and tracking records is especially important, because attrition 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 original data.[34] 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.[35]

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.[36] 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.[37] 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.[38]

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

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

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

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

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

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

[36] https://dimewiki.worldbank.org/ieddtab

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

[38] 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 single group of potentially eligible recipients into comparable groups of individuals who do and do not receive a treatment.[39] 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 the value of this variable at which eligibility stops.[40] Common examples are test score thresholds and income thresholds.[41] 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.[42] 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.[43] Similarly, spatial discontinuity designs are handled a bit differently due to their multidimensionality.[44]

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.[45] 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.[46] 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.[47]

[39] https://dimewiki.worldbank.org/Regression_Discontinuity

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

[41] https://blogs.worldbank.org/impactevaluations/regression-discontinuity-porn

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

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

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

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

[46] 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

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

In the analytical stage, regression discontinuity designs often include a large component of visual evidence presentation.[48] 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.[49] 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.[50] 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.

[48] https://faculty.smu.edu/kyler/courses/7312/presentations/baumer/Baumer_RD.pdf

[49] https://econ.lse.ac.uk/staff/spischke/ec533/RD.pdf

[50] 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 in a pattern such that its effect is not directly identifiable. Instead, similar to regression discontinuity designs, IV attempts to focus on a subset of the variation in treatment take-up and assesses that limited window of variation that can be argued to be unrelated to other factors.[51] To do so, the IV approach selects an **instrument** for the treatment status – an otherwise-unrelated predictor of exposure to treatment that affects the take-up status of an individual.[52] 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.

[51] 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

[52] https://dimewiki.worldbank.org/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.[53] 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.[54] IV designs furthermore rely on strong but untestable assumptions about the

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

[54] 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*

relationship between the instrument and the outcome.[55] 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 take-up of the treatment itself, especially in cases where take-up is expected to be low, or in circumstances where the treatment is available to those who are not specifically assigned to it ("encouragement designs").

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.[56] This can be done using the `weakiv` and `weakivtest` commands.[57] 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.[58] 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.[59]

[55] 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

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

[57] `https://www.carolinpflueger.com/WangPfluegerWeakivtest_20141202.pdf`

[58] 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`

[59] `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.[60] 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.

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

[60] `https://dimewiki.worldbank.org/Matching`

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**,[61] 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.[62] Newer methods, such as **coarsened exact matching**,[63] are designed to remove some of the dependence on 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.

[61] **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.

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

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

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.[64] The coarsened exact matching (`cem`) package applies the nonparametric approach.[65] DIME's `iematch` command in the `ietoolkit` package produces matchings based on a single continuous matching variable.[66] 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.

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

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

[66] `https://dimewiki.worldbank.org/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.[67] 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

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

counterfactual version of the treated unit using an average of the other units available.[68] 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.[69] 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. 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.[70]

[68] 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

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

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

Chapter 4: Sampling, randomization, and power

Sampling and randomized assignment 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. In quasi-experimental methods, sampling determines what populations the study will be able to make meaningful inferences about, and randomization analyses simulate counterfactual possibilities; what would have happened in the absence of the event. Demonstrating that sampling and randomization were taken into consideration before going to field lends credibility to any research study.

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. These analytical dimensions are particularly important in the initial phases of development research – typically conducted well before any actual field work occurs – and often have implications for feasibility, planning, and budgeting.

In this chapter, we first cover the necessary practices to ensure that random processes are reproducible. We next turn to how to implement sampling and randomized assignment, both for simple, uniform probability cases, and more complex designs, such as those that require clustering or stratification. We include code examples so the guidance is concrete and applicable. The last section discusses power calculations and randomization inference, and how both are important tools to critically and quantitatively assess different sampling and randomization designs and to make optimal choices when planning studies.

Random processes

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, 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.[1]

Any process that includes a random component is a random process, including sampling, randomization, power calculation simulations, and algorithms like bootstrapping. 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.[2]

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

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

Implementing random processes reproducibly in Stata

For statistical programming to be considered reproducible, it must be possible for the outputs of random processes can be re-obtained at a future time.[3] 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). At the end of the section, we provide a do-file that provides a concrete example of how to implement these principles.

[3] 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

Stata, like most statistical software, uses a **pseudo-random number generator**. Basically, it has a pre-calculated really long ordered list of numbers with the property that knowing the previous one gives you precisely zero information about the next one, i.e. a list of random numbers. Stata uses one number from this list 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.[4] 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: **versioning**, **sorting**, and **seeding**.

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

Versioning means using the same version of the software each time you run the random process. If anything is different, the underlying list of random numbers may have changed, and it will be impossible to recover the original result. In Stata, the `version` command ensures that the list of random numbers is fixed.[5] The `ieboilstart` command in `ietoolkit` provides functionality to support this requirement.[6] We recommend you use `ieboilstart` at the beginning of your master do-file.[7] However, testing your do-files

[5] At the time of writing we recommend using `version 13.1` for backward compatibility; the algorithm used to create this list of random numbers 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.

[6] https://dimewiki.worldbank.org/ieboilstart

[7] https://dimewiki.worldbank.org/Master_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 random 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. Since the exact order must be unchanged, the underlying data itself must be unchanged as well between runs. This means that if you expect the number of observations to change (for example increase during ongoing data collection) your randomization will not be stable unless you split your data up into smaller fixed datasets where the number of observations does not change. You can combine all those smaller datasets after your randomization. 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 in the list of random numbers. The seed is a number that should be at least six digits long and you should use exactly one unique, different, and randomly created seed per randomization process.[8] In Stata, `set seed [seed]` will set the generator to that start-point. In R, the `set.seed` function does the same. 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 should also describe in your code how the seed was selected. 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.[9]

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 your results are reproducible.

[8] You can draw a uniformly distributed six-digit seed randomly by visiting `https://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.

[9] `https://dimewiki.worldbank.org/Randomization_in_Stata`

replicability.do

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

* Load the auto dataset (auto.dta is a test dataset included in all Stata installations)
    sysuse auto.dta , clear

* SORTING - sort on the uniquely identifying variable "make"
    isid make, sort

* SEEDING - Seed picked using https://bit.ly/stata-random
    set seed 287608

* Demonstrate stability after VERSIONING, SORTING and SEEDING
    gen check1 = rnormal()  // Create random number
    gen check2 = rnormal()  // Create a second random number without resetting seed

    set seed 287608         // Reset the seed
    gen check3 = rnormal()  // Create a third random number after resetting seed

* Visualize randomization results. See how check1 and check3 are identical,
* but check2 is random relative check1 and check3
    graph matrix check1 check2 check3 , half
```

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 of the data indicating that results or correlations are true when they are not. In large studies, we can use what are called **asymptotic standard errors**[10] 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

[10] https://stats.stackexchange.com/q/88491

randomization.

Sampling

Sampling is the process of randomly selecting units of observation from a master list of individuals to be included in data collection.[11] 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 dataset**[12], creating an authoritative source for the existence and fixed characteristics of each of the units that may be surveyed.[13] The master dataset 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 dataset 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 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 it works. The do-file below provides an example of how to implement uniform-probability sampling in practice.

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

[12] https://dimewiki.worldbank.org/Master_Data_Set

[13] https://dimewiki.worldbank.org/Unit_of_Observation

simple-sample.do

```
* Set up reproducbilitiy - VERSIONING, SORTING and SEEDING
    ieboilstart , v(13.1)    // Version
    `r(version)'             // Version
    sysuse bpwide.dta, clear // Load data
    isid patient, sort       // Sort
    set seed 215597          // Seed - drawn using https://bit.ly/stata-random

* Generate a random number and use it to sort the observation. Then
* the order the observations are sorted in is random.
    gen sample_rand = rnormal() // Generate a random number
    sort sample_rand            // Sort based on the random number

* Use the sort order to sample 20% (0.20) of the observations. _N in
* Stata is the number of observations in the active dataset , and _n
* is the row number for each observation. The bpwide.dta has 120
* observations, 120*0.20 = 24, so (_n <= _N * 0.20) is 1 for observations
* with a row number equal to or less than 24, and 0 for all other
* observations. Since the sort order is randomized this mean that we
* have randomly assigned 20% of the sample.
    gen sample = (_n <= _N * 0.20)

* Restore the original sort order
    isid patient, sort

* Check your result
    tab sample
```

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 is 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 processes used for randomization are the same as those used for sampling, 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.[14] 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.

[14] 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.[15] Sampling typically has only two possible outcomes: observed and unobserved. Randomization, by contrast, often involves multiple possible results which each represent different varieties of treatments to be delivered; in some 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 before implementing it in code. The do-file below provides an example of how to implement a simple random assignment of multiple treatment arms.

[15] https://dimewiki.worldbank.org/Randomization_in_Stata

simple-multi-arm-randomization.do

```
* Set up reproducbilitiy - VERSIONING, SORTING and SEEDING
    ieboilstart , v(13.1)    // Version
    `r(version)'             // Version
    sysuse bpwide.dta, clear // Load data
    isid patient, sort       // Sort
    set seed 654697          // Seed - drawn using https://bit.ly/stata-random

* Generate a random number and use it to sort the observation. Then
* the order the observations are sorted in is random.
    gen treatment_rand = rnormal() // Generate a random number
    sort treatment_rand            // Sort based on the random number

* See simple-sample.do example for an explanation of "(_n <= _N * X)". The code
* below randomly selects one third of the observations into group 0, one third into group 1 and
* one third into group 2. Typically 0 represents the control group and 1 and
* 2 represents two treatment arms
    generate treatment = 0                      // Set all observations to 0
    replace  treatment = 1 if (_n <= _N * (2/3)) // Set only the first two thirds to 1
    replace  treatment = 2 if (_n <= _N * (1/3)) // Set only the first third to 2

* Restore the original sort order
    isid patient, sort

* Check your result
    tab treatment
```

Some types of experimental designs necessitate that randomization results be revealed in the field. It is possible to do this using survey software or live events, such as a live lottery. 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.[16] These techniques can be used in any random process; their implementation is nearly identical in both sampling and randomization.

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

Clustering

Many studies observe data at a different level than the randomization unit.[17] For example, a policy may be implemented at the village level, but the outcome of interest for the study is behavior changes at the household level. This type of structure is called **clustering**,[18] 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.

[17] https://dimewiki.worldbank.org/Unit_of_Observation

[18] https://dimewiki.worldbank.org/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 dataset where each cluster unit is an observation, randomize on that dataset, 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.[19] This accounts for the design covariance within the cluster – the information that if one individual

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

was observed or treated there, the other members of the clustering group were as well.

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.[20] 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 treatment arms. In this context, the subgroups are called **strata**.

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.[21] 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.[22] The user-written `randtreat` command properly implements stratification, as shown in the do-file below. 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.

[20] https://dimewiki.worldbank.org/Stratified_Random_Sample

[21] https://blogs.worldbank.org/impactevaluations/tools-of-the-trade-doing-stratified-randomization-with-uneven-numbers-in-some-strata

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

randtreat-strata.do

```
* If user written command randtreat is not installed, install it here
    cap which randtreat
    if _rc ssc install randtreat

* Set up reproducbilitiy - VERSIONING, SORTING and SEEDING
    ieboilstart , v(13.1)    // Version
    `r(version)'             // Version
    sysuse bpwide.dta, clear // Load data
    isid patient, sort       // Sort
    set seed 796683          // Seed - drawn using https://bit.ly/stata-random

* Create strata indicator. The indicator is a categorical variable with
* a different value for each unique combination of gender and age group.
    egen sex_agegroup = group(sex agegrp) , label
    label var sex_agegroup "Strata Gender and Age Group"

* Use the user written command randtreat to randomize when the groups cannot
* be evenly distributed into treatment arms. This is the case here, since
* there are 20 observations in each strata and 6 treatment arms to randomize
* them into. This will always result in two remaining ("misfits") observations
* in each group. randtreat offers different ways to deal with misfits. In this
* example, we use the "global" misfit strategy, meaning that the misfits will
* be randomized into treatment groups so that the sizes of the treatment
* groups are as balanced as possible globally (read the help file for details
* on this and other strategies for misfits). This way we have 6 treatment
* groups with exactly 20 observations in each, and it is randomized which
* group that has an extra observation in each treatment arm.
    randtreat,             ///
        generate(treatment)  /// New variable name
        multiple(6)          /// 6 treatment arms
        strata(sex_agegroup) /// Variable to use as strata
        misfits(global)      /// Misfit strategy if uneven groups

 * Label the treatment variable
    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

* Show result of randomization
    tab treatment sex_agegroup
```

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.[23] The exact formula depends on the analysis being performed, but is usually related to the inverse of the likelihood of inclusion.

[23] https://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.[24] 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 occurrence.[25] Power calculation and randomization inference are the two key tools to doing so.

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

[25] 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.[26] The purpose of power calculations is to identify where the strengths and weaknesses of your design are located, so you know the relative tradeoffs you will face by changing your randomization scheme for the final design. They also allow realistic interpretations of evidence: low-power studies can be very interesting, but they have a correspondingly higher likelihood of reporting false positive results.

[26] https://www.stat.columbia.edu/~gelman/stuff_for_blog/chap20.pdf

The classic definition of power is the likelihood that a design

detects a significant treatment effect, given that there is a non-zero treatment 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)**[27] 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 problem.

[27] https://dimewiki.worldbank.org/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.[28] 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 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.

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

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.

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. It is difficult to draw a firm line on when a study is "large enough", but even when a treatment effect is real, it can take ten or one hundred times more sample size than the minimum required power to ensure that the average statistically significant estimate of the treatment effect is accurate.[29] 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. We also don't need to: these methods were developed when it was very costly to compute large numbers of combinations using computers, and in most cases we now have the ability to do these calculations reasonably quickly.

[29] Schwarzer, G., Carpenter, J. R., and Rücker, G. (2015). Small-study effects in meta-analysis. In *Meta-analysis with R*, pages 107–141. Springer

Instead, we directly simulate a large variety of possible alternative randomizations. Specifically, the user-written `ritest` command[30] 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 could 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[31] significance levels may be very different than those given by asymptotic confidence intervals, particularly in small samples (up to several hundred clusters).

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

[31] `https://dimewiki.worldbank.org/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.

Chapter 5: Acquiring development data

Much of the recent push toward credibility in the social sciences has focused on analytical practices. However, credible development research often depends, first and foremost, on the quality of the raw data. When you are using original data - whether collected for the first time through surveys or sensors or acquired through a unique partnership - there is no way for anyone else in the research community to validate that it accurately reflects reality and that the indicators you have based your analysis on are meaningful. This chapter details the necessary components for a high-quality data acquisition process, no matter whether you are receiving large amounts of unique data from partners or fielding a small, specialized custom survey. By following these guidelines, you will be able to move on to data analysis, assured that your data has been obtained at high standards of both quality and security.

The chapter begins with a discussion of some key ethical and legal descriptions to ensure that you have the right to do research using a specific dataset. Particularly when confidential data is collected by you and your team or shared with you by a program implementer, government, or other partner, you need to make sure permissions are correctly granted and documented. Clearly establishing ownership and licensing of all information protects the privacy rights of the people it describes and your own right to publish. While the principles of data governance and data quality apply to all types of data, there are additional considerations to ensuring data quality if you are collecting data yourself through an instrument like a field survey. This chapter provides detailed guidance on the data generation workflow, from questionnaire design to programming electronic instruments and monitoring data quality. As there are many excellent resources on questionnaire design and field supervision, but few covering the particular challenges and opportunities presented by electronic surveys, we focus on the specific workflow considerations to ensure that digitally-generated data is accurate and usable in statistical software. There are many survey software options, and the market is rapidly evolving, so we focus on workflows and primary concepts rather than software-specific tools. We conclude with a discussion of safe handling, storage, and sharing of data. Secure file management is a basic requirement to comply with the legal and ethical agreements that allow access to personal information for research purposes.

Acquiring data

High-quality data is essential to most modern development research. Many research questions require the generation of novel data, because no source of reliable official statistics or other public data that addresses the inputs or outcomes of interest. Data generation can take many forms, including: primary data collection through surveys; private sector partnerships granting access to new data

sources, including administrative and sensor data; digitization of paper records, including administrative data; primary data capture by unmanned aerial vehicles or other types of remote sensing; or novel integration of various types of datasets, e.g. combining survey and sensor data. Except in the case of primary surveys funded by the research team, the data is typically not owned by the research team. Data ownership and licensing agreements are required for the research team to access the data and publish derivative work.

Data ownership

Data ownership is a tricky subject, as many jurisdictions have differing laws regarding data and information, and you may even be subject to multiple conflicting regulations. In some places, data is implicitly owned by the people who it is about. In others, it is owned by the people who collected it. In still more, it is highly unclear and there may be varying norms. The best approach is always to consult with a local partner and to make explicit agreements (including consent, where applicable) about who owns any data that is collected. Particularly where personal data, business data, or government data is involved – that is, when people are disclosing information to you that you could not obtain simply by walking around and looking – you should be extremely clear up front about what will be done with data so that there is no possibility of confusion down the line.

As with all forms of ownership, data ownership can include a variety of rights and responsibilities which can be handled together or separately. Ownership of data may or may not give you the right to share that data with other people, the right to publish or reproduce that data, or even the right to store data in certain ways or in certain locations. Again, clarity is key. If you are not collecting the data directly yourself – for example, if a government, company, or agency is doing it for you – make sure that you have an explicit agreement with them about who owns the resulting data and what the rights and responsibilities of the data collectors are, including reuse, storage, and retention or destruction of data.

Data licensing agreements

Data licensing is the formal act of giving some data rights to others while retaining ownership of a particular dataset. If you are not the owner of the dataset you want to analyze, you should enter into a licensing or terms-of-use agreement to access it for research purposes. Similarly, when you own a dataset, you must consider whether the data can be made accessible to other researchers, and what terms-of-use you require.

As a researcher, it is your responsibility to respect the rights of people who own data and people who are described in it; but it is also your responsibility to make sure that information is as available and accessible it can be. These twin responsibilities can and do come into tension, so it is important to be fully informed about what others are doing and to fully inform others of what you are doing. Writing down and agreeing to specific details is a good way of doing that.

If the research team requires access to existing data for novel research, terms of use should be agreed on with the data owner, typically through a data licensing agreement. Keep in mind that the data owner is likely not familiar with the research process, and therefore may be surprised at some of the things you want to do if you are not clear up front. You will typically want intellectual property rights to all research outputs developed used the data, a license for all uses of derivative works, including public distribution (unless ethical considerations contraindicate this). This is important to allow the research team to store, catalog, and publish, in whole or in part, either the original licensed dataset or datasets derived from the original. Make sure that the license you obtain from the data owner allows these uses, and that you consult with the owner if you foresee exceptions with specific portions of the data.

If the research team contracts a vendor to collect original data, contract terms must clearly stipulate that the research team owns the data and maintains full intellectual property rights. The contract should also explicitly stipulate that the contracted firm is responsible for protecting the confidentiality of the respondents, and that the data collection will not be delegated to any third parties or used by the firm or subcontractors for any purpose not expressly stated in the contract, before, during or after the assignment. The contract should also stipulate that the vendor is required to comply with ethical standards for social science research, and adhere to the specific terms of agreement with the relevant Institutional Review Board or applicable local authority.

Research teams that collect their own data must consider the terms under which they will release that data to other researchers or to the general public. Will you publicly release the data in full (removing personal identifiers)? Would you be okay with it being stored on servers anywhere in the world, even ones that are owned by corporations or governments in countries other than your own? Would you prefer to decide permission on a case-by-case basis, dependent on specific proposed uses? Would you expect that users of your data cite you or give you credit, or would you require them in turn to release their derivative data or publications under similar licenses as yours? Whatever your answers are to these questions,

make sure your license or other agreement under which you publish the data specifically details those requirements.

Receiving data from development partners

Research teams granted access to existing data may receive that data in a number of different ways. You may receive access to servers or accounts that already exist. You may receive a one-time transfer of a block of data, or you may be given access to a restricted area to extract information. In all cases, you must take action to ensure that data is transferred through secure channels so that confidential data is not compromised. Talk to an information-technology specialist, either at your organization or at the partner organization, to ensure that data is being transferred, received, and stored in a method that conforms to the relevant level of security. Keep in mind that compliance with ethical standards may in some cases require a stricter level of security than initially proposed by the partner agency.

Another important consideration at this stage is proper documentation and cataloging of data and associated metadata. It is not always clear what pieces of information jointly constitute a "dataset", and many of the sources you receive data from will not be organized for research. To help you keep organized and to put some structure on the materials you will be receiving, you should always retain the original data as received alongside a copy of the corresponding ownership agreement or license. You should make a simple "readme" document noting the date of receipt, the source and recipient of the data, and a brief description of what it is. All too often data produced by systems is provided as vaguely-named spreadsheets, or transferred as electronic communications with non-specific titles, and it is not possible to keep track of these kinds of information as data over time. Eventually, you will want to make sure that you are creating a collection or object that can be properly submitted to a data catalog and given a reference and citation. The metadata - documentation about the data - is critical for future use of the data. Metadata should include documentation of how the data was created, what they measure, and how they are to be used. In the case of survey data, this includes the survey instrument and associated manuals; the sampling protocols and field adherence to those protocols, and any sampling weights; what variable(s) uniquely identify the dataset(s), and how different datasets can be linked; and a description of field procedures and quality controls. We use as a standard the Data Documentation Initiative (DDI), which is supported by the World Bank's Microdata Catalog.[1]

[1] https://microdata.worldbank.org

As soon as the requisite pieces of information are stored together,

think about which ones are the components of what you would call a dataset. This is more of an art than a science: you want to keep things together that belong together, but you also want to keep things apart that belong apart. There usually won't be a precise way to answer this question, so consult with others about what is the appropriate level of aggregation for the data you have endeavored to obtain. This is the object you will think about cataloging, releasing, and licensing as you move towards the publication part of the research process. This may require you to re-check with the provider about what portions are acceptable to license, particularly if you are combining various datasets that may provide even more information about specific individuals.

Collecting data using electronic surveys

If you are collecting data directly from the research subjects yourself, you are most likely designing and fielding an electronic survey. These types of data collection technologies have greatly accelerated our ability to bring in high-quality data using purpose-built survey instruments, and therefore improved the precision of research. At the same time, electronic surveys create new pitfalls to avoid. Programming surveys efficiently requires a very different mindset than simply designing them in word processing software, and ensuring that they flow correctly and produce data that can be used in statistical software requires careful organization. This section will outline the major steps and technical considerations you will need to follow whenever you field a custom survey instrument, no matter the scale.

Developing a data collection instrument

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.[2] The focus of this section is the design of electronic field surveys, often referred to as Computer Assisted Personal Interviews (CAPI).[3] Although most surveys are now collected electronically, by tablet, mobile phone or web browser, **questionnaire design**[4] (content development) and questionnaire programming (functionality development) should be seen as two strictly separate tasks. Therefore, the research team should agree on all questionnaire content and design a paper version of the survey before beginning to program the electronic version.

[2] Glewwe, P. and Grosh, M. E. (2000). *Designing household survey questionnaires for developing countries: lessons from 15 years of the living standards measurement study.* World Bank

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

[4] https://dimewiki.worldbank.org/Questionnaire_Design

This facilitates a focus on content during the design process and ensures teams have a readable, printable version of their questionnaire. Most importantly, it means the research, not the technology, drives the questionnaire design.

We recommend this approach because an easy-to-read paper questionnaire is especially useful for training data collection staff, by focusing on the survey content and structure before diving into the technical component. It is much easier for enumerators to understand the range of possible participant responses and how to handle them correctly on a paper survey than on a tablet, and it is much easier for them to translate that logic to digital functionality later. Finalizing this version of the questionnaire before beginning any programming also avoids version control concerns that arise from concurrent work on paper and electronic survey instruments. Finally, a readable paper questionnaire is a necessary component of data documentation, since it is difficult to work backwards from the survey program to the intended concepts.

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.[5] It is essential to start with a clear understanding of the **theory of change**[6] and **experimental design** for your project. The first step of questionnaire design is to list key outcomes of interest, as well as the main covariates to control for and any variables needed for experimental design. The ideal starting point for this is a **pre-analysis plan**.[7]

Use the list of key outcomes to create an outline of questionnaire *modules*. Do not number the modules; instead use a short prefix so they can be easily reordered. For each module, determine if the module is applicable to the full sample, an 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.[8] At this point, it is useful to do a **content-focused pilot**[9] 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. Questionnaires must also include ways to document the reasons for **attrition** and treatment **contamination**. These are essential data components for completing CONSORT records, a standardized system for reporting enrollment, intervention allocation, follow-up,

[5] https://iriss.stanford.edu/sites/g/files/sbiybj6196/f/questionnaire_design_1.pdf

[6] https://dimewiki.worldbank.org/Theory_of_Change

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

[8] https://dimewiki.worldbank.org/Literature_Review_for_Questionnaire

[9] https://dimewiki.worldbank.org/Piloting_Survey_Content

and data analysis through the phases of a randomized trial.[10]

Once the content of the survey is drawn up, the team should conduct a small **survey pilot**[11] using the paper forms to finalize questionnaire design and detect any content issues. A content-focused pilot[12] is best done on pen and paper, before the questionnaire is programmed, because changes at this point may be deep and structural, which are hard to adjust in code. 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.[13] 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. Once the team is satisfied with the content and structure of the survey, it is time to move on to implementing it electronically.

[10] 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

[11] https://dimewiki.worldbank.org/Survey_Pilot

[12] https://dimewiki.worldbank.org/Piloting_Survey_Content

[13] https://dimewiki.worldbank.org/index.php?title=Checklist:_Refine_the_Questionnaire_(Content)

Designing surveys for electronic deployment

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 a software-specific form builder, all of which are accessible even to novice users.[14] 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.[15] 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 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 your analysis software.

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 fixed responses for the final version of the questionnaire – do not count on coding up lots of free text after a full survey. 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**[16]. 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

[14] https://dimewiki.worldbank.org/Questionnaire_Programming

[15] https://dimewiki.worldbank.org/SurveyCTO_Coding_Practices

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

(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 responses in the first few weeks of field work. Adding an answer category for a response frequently showing up as *other* can save time, as it avoids extensive post-coding.

It is essential to name the fields in your questionnaire in a way that will also work in your data analysis software. Most survey programs will not enforce this by default, since limits vary by software, and surveys will subtly encourage you to use long sentences and detailed descriptions of choice options. This is what you want for the enumerator-respondent interaction, but you should already have analysis-compatible labels programmed in the background so the resulting data can be rapidly imported in analytical software. There is some debate over how exactly 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.[17] Variable names should never include spaces or mixed cases (we prefer all-lowercase naming). Take special care with the length: very long names will be cut off in some softwares, which could result in a loss of uniqueness and lots of manual work to restore compatibility. We further discourage explicit question numbering, at least at first, as it discourages re-ordering questions, 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 uninformative variables names like `ag_15a`, `ag_15_new`, `ag_15_fup2`, and so on.

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

Programming electronic questionnaires

The starting point for questionnaire programming is therefore 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. Changing structural components of the survey after programming has been started often requires the

coder to substantially re-work the entire code. This is because the more efficient way to code surveys is non-linear. When programming, we do not start with the first question and proceed through to the last question. Instead, we 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, particularly where sections or field are interdependent or repeated in complex ways.

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:

- **Localizations**: the survey instrument should display full text questions and responses in the survey language, and it should also have English and code-compatible versions of all text and labels.
- **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.

All survey softwares 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,[18] part of the Stata package `iefieldkit`, to implement a form-checking routine for **SurveyCTO**, a proprietary implementation of the **Open Data Kit (ODK)** software. Intended for use during questionnaire programming and before field work, `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.

[18] https://dimewiki.worldbank.org/ietestform

A second survey pilot should be done after the questionnaire is programmed. The objective of this **data-focused pilot**[19] is to validate the 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.

[19] https://dimewiki.worldbank.org/index.php?title=Checklist:_Refine_the_Questionnaire_(Data)

Data quality assurance

Whether you are acquiring data from a partner or collecting it directly, it is important to make sure that data faithfully reflects

ground realities. Data quality assurance requires a combination of real-time data checks and back-checks or validation audits, which often means tracking down the people whose information is in the dataset.

Implementing high frequency quality checks

A key advantage of continuous electronic data intake methods, as compared to traditional paper surveys and one-time data dumps, is the ability to access and analyze the data while the project is ongoing. Data issues can be identified and resolved in real-time. Designing systematic data checks and running them routinely throughout data intake simplifies monitoring and improves data quality. As part of data collection preparation, the research team should develop a **data quality assurance plan**[20]. While data collection is ongoing, a research assistant or data analyst should work closely with the field team or partner to ensure that the data collection is progressing correctly, and set up and perform **high-frequency checks (HFCs)** with the incoming data.

[20] https://dimewiki.worldbank.org/Data_Quality_Assurance_Plan

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 effort is centered where it is most important. Data quality checks should be run on the data every time it is received from the field or partner to flag irregularities in the aquisition progress, in sample completeness, or in response quality. `ipacheck`[21] is a very useful command that automates some of these tasks, regardless of the source of the data.

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

It is important to check continuously that the observations in the data match the intended sample. In surveys, the software often provides some form of case management features through which sampled units are directly assigned to individual enumerators. For data recieved from partners, this may be harder to validate, since they are the authoritative source of the data, so cross-referencing with other data sources may be necessary to validate data. Even with careful management, it is often the case that raw data includes duplicate or missing entries, which may occur due to data entry errors or failed submissions to data servers.[22] `ieduplicates`[23] provides a workflow for collaborating on the resolution of duplicate entries between you and the provider. Then, 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 data collection progress is important for monitoring

[22] https://dimewiki.worldbank.org/Duplicates_and_Survey_Logs

[23] https://dimewiki.worldbank.org/ieduplicates

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 data collection completion rate and sample compliance by surveyor and survey team, if applicable, or compare data missingness across administrative regions, to identify any clusters that may be providing data of suspect quality.

High frequency checks should also include content-specific data checks. Electronic survey and data entry software often incorporates many quality control features, so these checks should focus on issues survey software cannot check automatically. As most of these checks are project specific, it is difficult to provide general guidance. An in-depth knowledge of the questionnaire and a careful examination of the analysis plan is the best preparation. Examples include verifying consistency across multiple response fields, validation of complex calculations like crop yields or medicine stocks (which require unit conversions), suspicious patterns in survey timing, or atypical response patters from specific data sources or enumerators.[24] Electronic data entry software typically provides rich metadata, which can be useful in assessing data quality. For example, automatically collected timestamps show when data was submitted and (for surveys) how long enumerators spent on each question, and trace histories show how many times answers were changed before or after the data was submitted.

[24] https://dimewiki.worldbank.org/Monitoring_Data_Quality

High-frequency checks will only improve data quality if the issues they catch are communicated to the data collection team. There are lots of ways to do this; what's most important is to find a way to create actionable information for your team. `ipacheck`, for example, generates a spreadsheet with flagged errors; these can be sent directly to the data collection teams. Many teams choose other formats to display results, such as 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 field work areas, whether field supervisors have access to laptops, internet speed, and coding skills of the team preparing the HFC workflows.

Conducting back-checks and data validation

Careful validation of data is essential for high-quality data. Since we cannot control natural measurement error that comes from variation in the realization of key outcomes, original data collection provides the opportunity to make sure that there is no error arising from

inaccuracies in the data itself. **Back-checks**[25] and other validation audits help ensure that data collection is following established protocols, and that data is not fasified, incomplete, or otherwise suspect. For back-checks and validation audits, a random subset of the main data is selected, and a subset of information from the full survey is verified through a brief targeted survey with the original respondent or a cross-referenced dataset from another source (if the original data is not a field survey). Design of the back-checks or validations follows the same survey design principles discussed above: you should use the analysis plan or list of key outcomes to establish which subset of variables to prioritize, and similarly focus on errors that would be major flags for poor quality data.

[25] https://dimewiki.worldbank.org/Back_Checks

Real-time access to the data massively increases the potential utility of validation, and both simplifies and improves the rigor of the associated workflows. You can use the raw data to draw the back-check or validation sample; this ensures that the validation is correctly apportioned across observations. As soon as checking is complete, the comparator data can be tested against the original data to identify areas of concern in real-time. The `bcstats` command is a useful tool for analyzing back-check data in Stata.[26] Some electronic surveys 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 enumerators are conducting interviews as expected. Do note, however, that audio audits must be included in the informed consent for the respondents.

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

Finalizing data collection

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 attrition occurred in the field and how they were implemented and resolved.

Collecting and sharing data securely

All confidential data must be handled in a way where there is no risk that anyone who is not approved by an Institutional Review Board (IRB)[27] for the specific project has the ability to access the data. Data can be confidential for multiple reasons, but the two most common reasons are that it contains personally identifiable information (PII)[28] or that the partner providing the data does not want it to be released.

[27] https://dimewiki.worldbank.org/IRB_Approval

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

Central to data security is **data encryption**, which is a group of methods that ensure that files are unreadable even if laptops are stolen, servers are hacked, or unauthorized access to the data is obtained in any other way.[29] Proper encryption is rarely just a single method, as the data will travel through many servers, devices, and computers from the source of the data to the final analysis. Encryption should be seen as a system that is only as secure as its weakest link. This section recommends a workflow with as few parts as possible, so that it is easy as possible to make sure the weakest link is still sufficiently secure.

[29] https://dimewiki.worldbank.org/Encryption

Encrypted data is made readable again using decryption, and decryption requires a password or a key. You must never share passwords or keys by email, WhatsApp or other insecure modes of communication; instead you must use a secure password manager.[30] In addition to providing a way to securely share passwords, password managers also provide a secure location for long term storage for passwords and keys, whether they are shared or not.

[30] https://lastpass.com or https://bitwarden.com

Many data-sharing software providers you are using will promote their services by saying they have on-the-fly encryption and decryption. While this is not a bad thing and it makes your data more secure, on-the-fly encryption is never secure enough. This is because, in order to make it automatic, the service provider needs to keep a copy of the password or key. Since it unlikely that that software provider is included in your IRB, this is not secure enough. This includes all file syncing software such as Dropbox, who, in addition to being able to view your data, may require you to give them additional legal usage rights in order to host it on their servers. It is possible, in some enterprise versions of data sharing software, to set up appropriately secure on-the-fly encryption. However, that setup is advanced, and you should never trust it unless you are a cybersecurity expert, or a cybersecurity expert within your organization has specified what it can be used for. In all other cases you should follow the steps laid out in this section.

Collecting data securely

In field surveys, most common data collection software will automatically encrypt all data in transit (i.e., upload from field or download from server).[31] If this is implemented by the software you are using, then your data will be encrypted from the time it leaves the device (in tablet-assisted data collection) or 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 that are used in data collection have a secure logon password and are never left unlocked.

[31] https://dimewiki.worldbank.org/Encryption#Encryption_in_Transit

Even though your data is therefore usually safe while it is being transmitted, it is not automatically secure when it is being stored. **Encryption at rest**[32] is the only way to ensure that confidential data remains private when it is stored on a server on the internet. You must keep your data encrypted on the data collection server whenever PII is collected, or when this is required by the data sharing agreement. If you do not, the raw data will be accessible by individuals who are not approved by your IRB, such as tech support personnel, server administrators, and other third-party staff. Encryption at rest must be used to make 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 decryption key is lost, there is absolutely no way to recover your data.

[32] https://dimewiki.worldbank.org/Encryption#Encryption_at_Rest

You should not assume that your data is encrypted at rest by default because of the careful protocols necessary. In most data collection platforms, encryption at rest needs to be explicitly enabled and operated by the user. There is no automatic way to implement this protocol, because the encryption key that is generated may never pass through the hands of a third party, including the data storage application. Most survey software implement **asymmetric encryption**[33] where there are two keys in a public/private key pair. Only the private key can be used to decrypt the encrypted data, and the public key can only be used to encrypt the data. It is therefore safe to send the public key to the tablet or the browser used to collect the data.

[33] https://dimewiki.worldbank.org/Encryption#Asymmetric_Encryption

When you enable encryption, the survey software will allow you to create and download – once – the public/private key pair needed to encrypt and decrypt the data. You upload the public key when you start a new survey, and all data collected using that public key can only be accessed with the private key from that specific

public/private key pair. You must store the key pair in a secure location, such as a password manager, as there is no way to access your data if the private key is lost. 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 key. Only project team members named in the IRB are allowed access to the private key.

Storing data securely

For most analytical needs, you typically need a to store the data somewhere other than the survey software's server, for example, on your computer or a cloud drive. While public/private key encryption is optimal for one-way transfer from the data collection device to the data collection server, it is not practical once you start interacting with the data. Instead, we use **symmetric encryption**[34] where we create a secure encrypted folder, using, for example, VeraCrypt.[35] Here, a single key is used to both encrypt and decrypt the information. Since only one key is used, the workflow can be simplified: the re-encryption after decrypted access can be done automatically, and the same secure folder can be used for multiple files. These files can be interacted with and modified like any unencrypted file as long as you have the key. The following workflow allows you to receive data and store it securely, without compromising data security:

[34] https://dimewiki.worldbank.org/Encryption#Symmetric_Encryption

[35] https://www.veracrypt.fr

1. Create a secure encrypted folder in your project folder. This should be on your computer, and could be in a shared folder.

2. Download data from the data collection server to that secure folder. If you encrypted the data during data collection, you will need *both* the private key used during data collection to be able to download the data, *and* you will need the key used when you created the secure folder to save it there. This your first copy of your raw data, and the copy you will use in your cleaning and analysis.

3. Create a secure folder on a flash drive or a external hard drive that you can keep in your office. Copy the data you just downloaded to this second secure folder. This is your "master" copy of your raw data. (Instead of creating a second secure folder, you can simply copy the first secure folder.)

4. Finally, create a third secure folder. Either you can create this on your computer and upload it to a long-term cloud storage

service, or you can create it on an external hard drive that you then store in a separate location, for example, at another office of your organization. This is your "golden master" copy of your raw data. You should never store the "golden master" copy of your raw data in a synced folder, where it is also deleted in the cloud storage if it is deleted on your computer. (Instead of creating a third secure folder, you can simply copy the first secure folder.)

This handling satisfies the **3-2-1 rule**: 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.[36] However, you still need to keep track of your encryption keys as without them your data is lost. If you remain lucky, you will never have to access your "master" or "golden master" copies – you just want to know it is there, safe, if you need it.

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

Sharing data securely

You and your team will use your first copy of the raw data as the starting point for data cleaning and analysis of the data. This raw dataset must remain encrypted at all times if it includes confidential data, which is almost always the case. As long as the data is properly encrypted, it can be shared using insecure modes of communication such as email or third-party syncing services. While this is safe from a data security perspective, this is a burdensome workflow, as anyone accessing the raw data must be listed on the IRB, have access to the decryption key and know how to use that key. Fortunately, there is a way to simplify the workflow without compromising data security.

To simplify the workflow, the confidential variables should be removed from your data at the earliest possible opportunity. This is particularly common in survey data, as identifying variables are often only needed during data collection. In this case, such variables may be removed as soon as the field work is completed, creating a de-identified copy of the data. Once the data is de-identified, it no longer needs to be encrypted – you and you team members can share it directly without having to encrypt it and handle decryption keys. The next chapter will discuss how to de-identify your data. This may not be so straightforward when access to the data is restricted by request of the data owner. If confidential information is 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.

The data security standards that apply when receiving confidential data also apply when transferring confidential data. A common example where this is often forgotten involves sharing survey information, such as sampling lists, with a field partner. This data is –

by all definitions – also PII data and must be encrypted. A sampling list can often be used to reverse identify a de-identified dataset, so if you were to share it using an insecure method, then that would be your weakest link that could render useless all the other steps you have taken to ensure the privacy of the respondents.

In some survey software, you can use the same encryption that allows you to receive data securely from the field, to also send data, such a sampling list, to the field. But if you are not sure how that is done, or even can be done, in the survey software you are using, then you should create a secure folder using, for example, VeraCrypt and share that secure folder with the field team. Remember that you must always share passwords and keys in a secure way like password managers.

At this point, the raw data securely stored and backed up. It can now be transformed into your final analysis dataset, through the steps described in the next chapter. Once the data collection is over, you typically will no longer need to interact with the identified data. So you should create a working version of it that you can safely interact with. This is described in the next chapter as the first task in the data cleaning process, but it's useful to get it started as soon as encrypted data is downloaded to disk.

Chapter 6: Analyzing research 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 typically a back-and-forth discussion between people with differing skill sets. An essential part of the process is translating the raw data received from the field into economically meaningful indicators. To effectively do this in a team environment, data, code and outputs must be well-organized, with a clear system for version control, and analytical scripts structured such that any member of the research team can run them. Putting in time upfront to structure data work well pays substantial dividends throughout the process.

In this chapter, we first cover data management: how to organize your data work at the start of a project so that coding the analysis itself is straightforward. This includes setting up folders, organizing tasks, master scripts, and putting in place a version control system so that your work is easy for all research team members to follow, and meets standards for transparency and reproducibility. Second, we turn to de-identification, a critical step when working with any personally-identified data. In the third section, we offer detailed guidance on data cleaning, from identifying duplicate entries to labeling and annotating raw data, and how to transparently document the cleaning process. Section four focuses on how to transform your clean data into the actual indicators you will need for analysis, again emphasizing the importance of transparent documentation. Finally, we turn to analysis itself. We do not offer instructions on how to conduct specific analyses, as that is determined by research design; rather, we discuss how to structure analysis code, and how to automate common outputs so that your analysis is fully reproducible.

Managing data effectively

The goal of data management is to organize the components of data work so the complete process can traced, understood, and revised without massive effort. We focus on 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. A **master script** connects folder structure and code. It is a one-file summary of your whole project. Finally, **version control** gives you clear file histories and backups, which enable the team to edit files without fear of losing information and track how each edit affects other files in the project.

Organizing your folder structure

There are many ways to organize research data. Our team at DIME Analytics developed the `iefolder`[1] command (part of `ietoolkit`[2]) to automatize the creation of folders following our preferred scheme and to standardize folder structures across teams and projects. A standardized structure greatly reduces the costs that PIs and RAs face when switching between projects, because folders are organized in exactly the same way and use the same file paths, shortcuts, and macro references.[3] We created `iefolder` based on our experience with survey data, but it can be used for other types of data. Other teams may prefer a different scheme, but the principle of creating a single unified standard remains.

[1] https://dimewiki.worldbank.org/iefolder

[2] https://dimewiki.worldbank.org/ietoolkit

[3] https://dimewiki.worldbank.org/DataWork_Folder

At the top level of the structure created by `iefolder` are what we call "round" folders.[4] You can think of a "round" as a single source of data, which will all be cleaned using a single script. Inside each round folder, 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**[5] so the structure of all project code is reflected in a top-level script.

[4] https://dimewiki.worldbank.org/DataWork_Survey_Round

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

Breaking down tasks

We divide the process of transforming raw datasets to research outputs into four steps: de-identification, data cleaning, variable construction, and data analysis. Though they are frequently implemented concurrently, creating separate scripts and datasets prevents mistakes. It will be easier to understand this division as we discuss what each stage comprises. What you should know for 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 dataset. The names of codes, datasets and outputs for each stage should be consistent, making clear how they relate to one another. So, for example, a script called `section-1-cleaning` would create a dataset called `section-1-clean`.

The division of a project in stages also facilitates a 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, in which team members read and run each other's codes. Reviewing code at each stage, rather than waiting until the end of a project, is preferrable as the amount of code to review is more manageable and it allows

you to correct errors in real-time (e.g. correcting errors in variable construction before analysis begins). Code review is a common quality assurance practice among data scientists. It helps to keep the quality of the outputs high, and is also a great way to learn and improve your own code.

Writing master scripts

Master scripts allow users to execute all the project code from a single file. As discussed in Chapter 2, the master script should briefly describe what each section of the code does, and map the files they require and create. The master script also connects code and folder structure through macros or objects. In short, a master script is a human-readable map of the tasks, files, and folder structure that comprise a project. Having a master script eliminates the need for complex instructions to replicate results. Reading it 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.

Implementing version control

Establishing a version control system is an incredibly useful and important step for documentation, collaboration and conflict-solving. 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. The focus in version control is often code, but changes to analysis outputs should, when possible, be version controlled together with the code edits. This way you know which edits in the code led to which changes in the outputs. If you are writing code in Git or 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 datasets, 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 datasets and results that they entail.

De-identifying research data

The starting point for all tasks described in this chapter is the raw dataset, which should contain the exact data received, with

no changes or additions. The raw data will invariably come in a variety of file formats and these files should be saved in the raw data folder *exactly as they were received*. Be mindful of how and where they are stored as they cannot be re-created and nearly always contain confidential data such as **personally-identifying information**. As described in the previous chapter, confidential data must always be encrypted[6] and be properly backed up since every other data file you will use is created from the raw data. The only datasets that can not be re-created are the raw data themselves.

[6] https://dimewiki.worldbank.org/Encryption

The raw data files should never be edited directly. This is true even in the rare case when the raw data cannot be opened due to, for example, incorrect encoding where a non-English character is causing rows or columns to break at the wrong place when the data is imported. In this scenario, you should create a copy of the raw data where you manually remove the special characters and securely back up *both* the broken and the fixed copy of the raw data. You will only keep working from the fixed copy, but you keep both copies in case you later realize that the manual fix was done incorrectly.

The first step in the transformation of raw data to an analysis-ready dataset is de-identification. This simplifies workflows, as once you create a de-identified version of the dataset, you no longer need to interact directly with the encrypted raw data. at this stage, means stripping the dataset of personally identifying information.[7] To do so, you will need to identify all variables that contain identifying information.[8] For data collection, where the research team designs the survey instrument, flagging all potentially identifying variables in the questionnaire design stage simplifies the initial de-identification process. If you did not do that, or you received original data by another means, there are a few tools to help flag variables with personally-identifying data. JPAL's `PII scan`, as indicated by its name, scans variable names and labels for common string patterns associated with identifying information.[9] The World Bank's `sdcMicro` lists variables that uniquely identify observations, as well as allowing for more sophisticated disclosure risk calculations.[10] The `iefieldkit` command `iecodebook` lists all variables in a dataset and exports an Excel sheet where you can easily select which variables to keep or drop.[11]

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

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

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

[10] https://sdctools.github.io/sdcMicro/articles/sdcMicro.html

[11] https://dimewiki.worldbank.org/Iecodebook

Once you have a list of variables that contain confidential information, assess them against the analysis plan and first ask yourself for each variable: *will this variable be needed for the analysis?* If not, the variable should be dropped. Don't be afraid to drop too many variables the first time, as you can always go back and remove variables from the list of variables to be dropped, but you can not go back in time and drop a PII variable that was leaked because it was incorrectly

kept. Examples include respondent names and phone numbers, enumerator names, taxpayer numbers, and addresses. For each confidential variable that is needed in the analysis, ask yourself: *can I encode or otherwise construct a variable that masks the confidential component, and then drop this variable?* This is typically the case for most identifying information. Examples include geocoordinates (after constructing measures of distance or area, drop the specific location) and names for social network analysis (can be encoded to secret and unique IDs). If the answer to either of the two questions above is yes, 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. If confidential information strictly required for the analysis itself and can not be masked or encoded, it will be necessary to keep at least a subset of the data encrypted through the data analysis process.

The resulting de-identified data will be the underlying source for all cleaned and constructed data. This is the dataset that you will interact with directly during the remaining tasks described in this chapter. Because identifying information is typically only used during data collection, when teams need to find and confirm the identity of interviewees, de-identification should not affect the usability of the data.

Cleaning data for analysis

Data cleaning is the second stage in the transformation of raw data into data that you can analyze. The cleaning process involves (1) making the dataset easy to use and understand, and (2) documenting individual data points and patterns that may bias the analysis. The underlying data structure does not change. The cleaned dataset 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. You need to acquire an extensive understanding of the contents and structure of the raw data. Explore the dataset using tabulations, summaries, and descriptive plots. Knowing your dataset well will make it possible to do analysis.

Identifying the identifier

The first step in the cleaning process is to understand the level of observation in the data (what makes a row), and what variable or set of variables uniquely identifies each observations. Ensuring

that observations are uniquely and fully identified[12] is possibly the most important step in data cleaning. It may be the case that the variable expected to be the unique identifier in fact is either incomplete or contains duplicates. This could be due to duplicate observations or errors in data entry. It could also be the case that there is no identifying variable, or the identifier is a long string, such as a name. In this case cleaning begins by carefully creating a variable that uniquely identifies the data. As discussed in the previous chapter, checking for duplicated entries is usually part of data quality monitoring, and is ideally addressed as soon as data is received

[12] https://dimewiki.worldbank.org/ID_Variable_Properties

Note that while modern survey tools create unique identifiers for each submitted data record, that is not the same as having a unique ID variable for each individual in the sample. You want to make sure the dataset has a unique ID variable that can be cross-referenced with other records, such as the master dataset[13] and other rounds of data collection. `ieduplicates` and `iecompdup`, two Stata commands included in the `iefieldkit` package,[14] create an automated workflow to identify, correct and document occurrences of duplicate entries.

[13] https://dimewiki.worldbank.org/Master_Data_Set

[14] https://dimewiki.worldbank.org/iefieldkit

Labeling, annotating, and finalizing clean data

The last step of data cleaning is to label and annotate the data, so that all users have the information needed to interact with it. There are three key steps: renaming, labeling and recoding. 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 easier.[15]

[15] https://dimewiki.worldbank.org/iecodebook

First, **renaming**: for data with an accompanying survey instrument, it is useful to keep the same variable names in the cleaned dataset as in the survey instrument. That way it's straightforward to link variables to the relevant survey question. Second, **labeling**: applying labels makes it easier to understand your data as you explore it, and thus reduces the risk of small errors making their way through into the analysis stage. Variable and value labels should be accurate and concise.[16]

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

Third, **recoding**: survey codes for "Don't know", "Refused to answer", and other non-responses must be removed but records of them should still be kept. In Stata that can elegantly be done using extended missing values.[17] String variables that correspond to categorical variables should be encoded. Open-ended responses stored as strings usually have a high risk of being identifiers, so they should be encoded into categories as much as possible and raw data points dropped. You can use the encrypted data as an input to a

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

construction script that categorizes these responses and merges them to the rest of the dataset.

Preparing a clean dataset

The main output of data cleaning is the cleaned dataset. It should contain the same information as the raw dataset, with identifying variables and data entry mistakes removed. Although original data typically requires more extensive data cleaning than secondary data, you should carefully explore possible issues in any data you are about to use. When reviewing raw data, you will inevitably encounter data entry mistakes, such as typos and inconsistent values. These mistakes should be fixed in the cleaned dataset, and you should keep a careful record of how they were identified, and how the correct value was obtained.[18]

[18] https://dimewiki.worldbank.org/Data_Cleaning

The cleaned dataset should always be accompanied by a dictionary or codebook. Survey data should be easily traced back to the survey instrument. Typically, one cleaned dataset will be created for each data source or survey instrument; and each row in the cleaned dataset represents one respondent or unit of observation.[19]

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

If the raw dataset is very large, or the survey instrument is very complex, you may want to break the data cleaning into sub-steps, and create intermediate cleaned datasets (for example, one per survey module). When dealing with complex surveys with multiple nested groups, is is also useful to have each cleaned dataset at the smallest unit of observation inside a roster. This will make the cleaning faster and the data easier to handle during construction. But having a single cleaned dataset will help you with sharing and publishing the data.

Finally, any additional information collected only for quality monitoring purposes, such as notes and duration fields, can also be dropped. To make sure the cleaned dataset 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 dataset and the documentation to support it, you have created the first data output of your project: a publishable dataset. 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 this dataset for publication, even if it will remain embargoed for some time. This will help you organize your files and create a backup of the data, and some donors require that the data be filed as an intermediate step of the project.

Documenting data cleaning

Throughout the data cleaning process, you will often need extensive inputs from the people responsible for data collection. (This could be a survey team, the government ministry responsible for administrative data systems, the technology firm that generated remote sensing data, etc.) You should acquire and organize all documentation of how the data was generated, such as reports from the data provider, field protocols, data collection manuals, survey instruments, supervisor notes, and data quality monitoring reports. These materials are essential for data documentation.[20] They should be stored in the corresponding `Documentation` folder for easy access, as you will probably need them during analysis, and should be published along with the data.

[20] https://dimewiki.worldbank.org/Data_Documentation

Include in the `Documentation` folder records of any corrections made to the data, including to duplicated entries, as well as communications where theses issues are reported. Be very careful not to include confidential information in documentation that is not securely stored, or that you intend to release as part of a replication package or data publication.

Another important component of data cleaning documentation are the results of data exploration. As you clean your dataset, take the time to explore the variables in it. Use tabulations, summary statistics, 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. Create a record of what you observe, then use it as a basis for discussions of how to address data issues during variable construction. This material will also be valuable during exploratory data analysis.

Constructing analysis datasets

The third stage is construction of the dataset you will use for analysis. It is at this stage that the cleaned data is transformed into analysis-ready data, by integrating different datasets and 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. During this process, the data points will typically be reshaped and aggregated so that level of the dataset goes from the unit of observation in the survey to the unit of analysis.[21]

[21] https://dimewiki.worldbank.org/Unit_of_Observation

A constructed dataset 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 datasets, depending on how your analysis is structured. Don't worry if you cannot create a single, "canonical" analysis dataset. 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 dataset.

Fitting construction into the data workflow

Construction is done separately from data cleaning for two reasons. First, it clearly differentiates correction of data entry errors (necessary for all interactions with the data) from creation of analysis indicators (necessary only for the specific analysis). Second, it ensures 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 require at least two cleaning scripts, and a single one for construction.

Construction of the analysis data should be done right after data cleaning and before data analysis starts, according to the pre-analysis plan. In practice, however, as you analyze the data, different constructed variables may become necessary, as well as subsets and other alterations to the data, and you will need to adjust the analysis data accordingly. Even if construction and analysis are done concurrently, you should always do the two in separate scripts. If every script that creates a table starts by loading a dataset, 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. Doing all variable construction in a single, separate script helps avoid this and ensure consistency across different outputs.

Integrating different data sources

Often, you will combine or merge information from different data sources together in order to create the analysis dataset. For example, you may merge administrative data with survey data to include

demographic information in your analysis, or you may want to integrate geographic information in order to construct indicators or controls based on the location of observations. To do this, you will need to consider the unit of observation for each dataset, and the identifying variable, to understand how they can be merged.

If the datasets you need to join have the same unit of observation, merging may be straightforward. The simplest case is merging datasets at the same unit of observation which use a consistent, uniquely and fully identifying ID variable. For example, in the case of a panel survey for firms, you may merge baseline and endline data using the firm identification number. In many cases, however, datasets at the same unit of observation may not use a consistent numeric identifier. Identifiers that are string variables, such as names, often contain spelling mistakes or irregularities in capitalization, spacing or ordering. In this case, you will need to do a **fuzzy match**, to link observations that have similar identifiers. In these cases, you will need to extensively analyze the merging patterns and understand what units are present in one dataset but not the other, as well as be able to resolve fuzzy or imperfect matching. There are some commands such as `reclink` in Stata that can provide some useful utilities, but often a large amount of close examination is necessary in order to figure out what the matching pattern should be and how to accomplish it in practice through your code.

In other cases, you will need to join data sources that have different units of observation. For example, you might be overlaying road location data with household data, using a spatial match, or combining school administrative data, such as attendance records and test scores, with household demographic characteristics from a survey. Sometimes these cases are conceptually straightforward. For example, merging a dataset of health care providers with a dataset of patients comes with a clear linking relation between the two; the challenge usually occurs in correctly defining statistical aggregations if the merge is intended to result in a dataset at the provider level. However, other cases may not be designed with the intention to be merged together, such as a dataset of infrastructure access points, for example, water pumps or schools and a dataset of household locations and roads. In those cases, a key part of the research contribution is figuring out what a useful way to combine the datasets is. Since these conceptual constructs are so important and so easy to imagine different ways to do, it is especially important that these data integrations are not treated mechanically and are extensively documented separately from other data construction tasks.

Integrating different datasets may involve changing the structure of the data, e.g. changing the unit of observation through collapses or

reshapes. This should always be done with great care. 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 different than what you expect. If you are subsetting your data, drop observations explicitly, indicating why you are doing that and how the data set changed.

Constructing analytical variables

Once you have assembled your different data sources, it's time to create the specific indicators of interest for analysis. New variables should be assigned functional names, and the dataset ordered such that related variables are together. Adding notes to each variable will make your dataset more user-friendly.

Before constructing new variables, you must check and double-check the value-assignments of questions, as well as the units and scales. This is when you will use the knowledge of the data and the documentation you acquired during cleaning. First, check that all categorical variables have the same value assignment, i.e., that labels and levels have the same correspondence across variables that use the same options. For example, it's possible that in one question `0` means "no" and `1` means "yes", while in another one the same answers were coded as `1` and `2`. (We recommend coding binary questions as either `1` and `0` or `TRUE` and `FALSE`, so they can be used numerically as frequencies in means and as dummies in regressions. Note that this implies re-expressing categorical variables like `sex` to binary variables like `woman`.) Second, 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 two acres and get a meaningful number.

You will also need to decide how to handle any outliers or unusual values identified during data cleaning. How to treat outliers is a question for the research team (as there are multiple possible approaches), but make sure to note what decision was made and why. Results can be sensitive to the treatment of outliers, so keeping the original variable in the dataset will allow you to test how much it affects the estimates. These points also apply to imputation of missing values and other distributional patterns.

Finally, creating a panel with survey 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 cleaned panel dataset, ignoring the previous constructed version of the baseline data. The `iecodebook append` subcommand will help you reconcile and append the cleaned survey rounds. After that, adapt a single variable construction script so it can be used on the panel dataset as a whole. In addition to preventing inconsistencies, this process will also save you time and give you an opportunity to review your original code.

Documenting variable 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 the `iecodebook export` subcommand to list all variables in the dataset, and complement it with the variable definitions you wrote during construction to create a concise metadata 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 datasets, 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.

Writing data analysis code

When data is cleaned and indicators constructed, you are ready to generate analytical outputs. There are many existing resources for data analysis, such as *R for Data Science*;[22] *A Practical Introduction*

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

to Stata;[23] *Mostly Harmless Econometrics*;[24] and *Causal Inference: The Mixtape*.[25] We focus on how to *code* data analysis, rather than how to conduct specific analyses.

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

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

[25] https://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. 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 dataset before each analysis task. To avoid mistakes, it's important to take the time to organize the code that you want to use again in a clean manner.

A well-organized analysis script starts with a completely fresh workspace and explicitly loads data before analyzing it, for each output it creates. This setup 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 require manual instructions for all necessary chunks of code to be run in the right order. Each chunk of analysis code should run completely independently of all other code, except for the master script. You could go as far as coding every output in a separate script (although you usually won't).

There is nothing wrong with code files being short and simple. In fact, analysis scripts should be as simple as possible, so whoever is reading them can focus on the econometrics, not the coding. All research questions and statistical decisions should be very explicit in the code, and should be very easy to detect from the way the code is written. This includes clustering, sampling, and control variables, to name a few. If you have multiple analysis datasets, 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.

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.

Visualizing data

Data visualization[26] is increasingly popular, and is becoming a field in its own right.[27] 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".[28] *Fundamentals of Data Visualization* provides extensive details on practical application;[29] as does *Data Visualization: A Practical Introduction*.[30] 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.[31] 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.[32] If you are an R user, the *R Graphics Cookbook*[33] is a great resource for the its popular visualization package `ggplot`[34]. But there are a variety of other visualization packages, such as `highcharter`,[35] `r2d3`,[36] `leaflet`,[37] and `plotly`,[38] to name a few. We have no intention of creating an exhaustive list, but this is a good place to start.

We attribute some of the difficulty of creating good data visualization to 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 we created the **Stata Visual Library**[39], which has examples of graphs created in Stata and curated by us.[40] The Stata Visual Library includes example datasets to use with each do-file, so you get a good sense of what your data should look like before you can start writing code to create a visualization.

Exporting analysis outputs

Our team has created a few products to automate common outputs and save you precious research time. The `ietoolkit` package includes

[26] https://dimewiki.worldbank.org/Data_visualization

[27] 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

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

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

[30] http://socvis.co

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

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

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

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

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

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

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

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

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

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

two commands to export nicely formatted tables. `iebaltab`[41] creates and exports balance tables to excel or LaTeX. `ieddtab`[42] does the same for difference-in-differences regressions. It also includes a command, `iegraph`,[43] to export pre-formatted impact evaluation results graphs.

It's okay 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.[44] 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 up 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 totally unnecessary. There are numerous commands to export outputs from both R and Stata to a myriad of formats.[45] 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 or 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, 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 so 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 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,

[41] https://dimewiki.worldbank.org/iebaltab

[42] https://dimewiki.worldbank.org/ieddtab

[43] https://dimewiki.worldbank.org/iegraph

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

[45] Some examples are http://repec.sowi.unibe.ch/stata/estoutestout, https://www.princeton.edu/~otorres/Outreg2.pdfoutreg2, and https://www.benjaminbdaniels.com/stata-code/outwriteoutwrite in Stata, and https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdfstargazer and https://ggplot2.tidyverse.org/reference/ggsave.htmlggsave in R.

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.[46]

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 dataset 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 the data cannot due to confidentiality. 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.

[46] https://dimewiki.worldbank.org/Checklist:_Reviewing_Graphs and https://dimewiki.worldbank.org/Checklist:_Submit_Table

Chapter 7: Publishing collaborative research

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 used 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. Publication typically involves multiple iterations of manuscript, code, and data files, with inputs from multiple collaborators. This process can quickly become unwieldy. It is in nobody's interest for a skilled and busy researcher to spend days re-numbering references (and it can take days) when a small amount of up-front effort can automate the task.

In this chapter, we suggest tools and workflows for efficiently managing collaboration and ensuring reproducible outputs. First, we discuss how to use dynamic documents to collaborate on technical writing. Second, we provide guidelines for preparing a functioning and informative replication package. If you have organized your analytical work according to the general principles outlined in earlier chapters, preparing to release materials will not require substantial reorganization of the work you have already done. Hence, this step represents the conclusion of the system of transparent, reproducible, and credible research we introduced from the very first chapter of this book. We include specific guidance on publishing both code and data files, noting that these can be a significant contribution in addition to analytical results. 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

Development economics research is increasingly a collaborative effort. This reflects changes in the economics discipline overall: the number of sole-authored papers is decreasing, and the majority of recent papers in top journals have three or more authors.[1] As a consequence, manuscripts typically pass back and forth between several writers before they are ready for publication, Just as with the preparation of analytical outputs, effective collaboration requires the adoption of tools and practices that enable version control and simultaneous contribution. **Dynamic documents** are a way to significantly simplify workflows: updates to the analytical outputs that appear in these documents, such as tables and figures, can be passed on to the final output with a single process, rather than copy-

[1] https://voxeu.org/article/growth-multi-authored-journal-articles-economics

and-pasted or otherwise handled individually. Managing the writing process in this way improves organization and reduces error, such that there is no risk of materials being compiled with out-of-date results, or of completed work being lost or redundant.

Preparing 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 way, updates will never be accidentally excluded, and 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.

The most widely utilized software for dynamically managing both text and results is LaTeX(pronounced "lah-tek").[2] LaTeXis a document preparation and typesetting system with a unique syntax. While this tool has a significant learning curve, its enormous flexibility in terms of operation, collaboration, output formatting, and styling make it the primary choice for most large technical outputs. In fact, LaTeXoperates behind-the-scenes in many other dynamic document tools (discussed below). Therefore, we recommend that you learn to use LaTeXdirectly as soon as you are able to and provide several resources for doing so in the next section.

There are tools that can generate dynamic documents from within your scripts, such as R's RMarkdown[3] Stata offers a built-in package for dynamic documents, `dyndoc`[4], and user-written commands such `texdoc`[5] and `markstat`[6] allow for additional functionalities. These tools "knit" or "weave" text and code together, and are programmed to insert code outputs in pre-specified locations. Documents called

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

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

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

[5] http://repec.sowi.unibe.ch/stata/texdoc

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

"notebooks" (such as Jupyter[7]) work similarly, as they also use the underlying analytical software to create the document. These tools are usually appropriate for short or informal documents because it tends to be difficult to edit the content unless using the tool and often does not have as extensive formatting option as, for example, Word.

[7] https://jupyter.org

There are also simple tools for dynamic documents that do not require direct operation of the underlying code or software, simply access to the updated outputs. 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. They have limited functionality in terms of version control and formatting, and may never include any references to confidential data, but can be useful for working on informal outputs, such as blogposts, with collaborators who do not code.

Technical writing with LaTeX

LaTeXis 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. LaTeXenables 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 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 LaTeXis the BibTeX citation and bibliography manager.[8] 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, LaTeXinserts references in text using the `\cite{}` command. Once this is written, LaTeXautomatically pulls all the citations into text and creates a complete bibliography based on the citations you used whenever you compile the document.

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

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 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.[9]

LaTeXhas one more useful trick: using **`pandoc`**,[10] you can translate the raw document into Word (or a number of other formats) by running the following code from the command line:

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

[10] https://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[11] 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.

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

Unfortunately, despite these advantages, LaTeXcan 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 LaTeXrequires 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[12] and `atom-latex`[13] offer the most flexibility to work with LaTeXon your computer, such as advanced integration with Git, the entire group of writers needs to be comfortable with LaTeXbefore adopting one of these tools. They can require a lot of troubleshooting at a basic level at first, and staff not used to programming may not be willing or able to acquire the necessary knowledge. Cloud-based implementations of LaTeX, discussed in the next section, allow teams to take advantage of the features of LaTeX, without requiring knowledge of the technical details.

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

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

Getting started with LaTeXin the cloud

LaTeXis 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 LaTeXwithout all members having to invest in new skills, we suggest using a cloud-based implementation as your first foray into LaTeXwriting. 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 LaTeXhave several advantageous features. First, since they are completely hosted online, they avoid the inevitable troubleshooting required to set up a LaTeXinstallation on various personal computers run by the different members of a team. Second, they 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. Third, they typically allow collaborators to edit documents simultaneously, though different services vary the number of collaborators and documents allowed at each tier. Fourth, and most usefully, some implementations 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 LaTeXcoding. 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.

Cloud-based implementations of LaTeXalso have disadvantages. There is still some up-front learning required, unless you're using the rich text editor. Continuous access to the Internet is necessary, and updating figures and tables requires a bulk file upload that is tough to automate. 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.[14] 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 and data should exactly reproduce the raw outputs you have used for the paper, and the replication file should not include any 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.

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

Publishing data for replication

Publicly documenting all original data generated as part of a research project is an important contribution in its own right. Publishing original datasets is a significant contribution that can be made in addition to any publication of analysis results.[15] If you are not able to publish the data itself, due to licensing agreements or ethical concerns, there are often options to catalog or archive the data. 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, and provide information about the contents of the datasets and how future users might request

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

permission to access them (even if you are not the person to grant that permission). They can also provide for timed future releases of datasets once the need for exclusive access has ended.

Publishing data allows other researchers to validate the mechanical construction of your results, investigate what other results might be obtained from the same population, and test alternative approaches or answer other questions. This fosters collaboration and may enable your team to fully explore variables and questions that you may not have time to focus on otherwise. There are different options for data publication. The World Bank's Development Data Hub[16] includes a Microdata Catalog[17] and a Geospatial Catalog, where researchers can publish data and documentation for their projects.[18] The Harvard Dataverse[19] publishes both data and code. The Datahub for Field Experiments in Economics and Public Policy[20] is especially relevant for impact evaluations. Both the World Bank Microdata Catalog and the Harvard Dataverse create data citations for deposited entries. DIME has its own collection of datasets in the Microdata Catalog, where data from our projects is published.[21]

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 still have the right to release derivate datasets, even if it is just the indicators you constructed and their documentation.[22] 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. 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.

When you do publish data, you decide how it may be used and what, if any license, you will assign to it.[23] Terms of use available in the World Bank Microdata Catalog include, in order of increasing restrictiveness: open access, direct access, and licensed access.[24] Open Access data is freely available to anyone, and simply requires attribution. Direct Access data is to registered users who agree to use the data for statistical and scientific research purposes only, to cite the data appropriately, and to not attempt to identify respondents or data providers or link to other datasets that could allow for re-identification. Licensed access data is restricted to bona fide users, who submit a documented application for how they will use the data and sign an agreement governing data use. The user must be acting on behalf of an organization, which will be held responsible in the case of any misconduct. 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 licensing rights is at the time that data collection or sharing agreements are signed.

Published data should be released in a widely recognized format.

[16] https://datacatalog.worldbank.org

[17] https://microdata.worldbank.org

[18] https://dimewiki.worldbank.org/Microdata_Catalog https://dimewiki.worldbank.org/Checklist:_Microdata_Catalog_submission

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

[20] https://dataverse.harvard.edu/dataverse/DFEEP

[21] https://microdata.worldbank.org/catalog/dime

[22] https://guide-for-data-archivists.readthedocs.io

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

[24] https://microdata.worldbank.org/index.php/terms-of-use

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 can be used 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. If possible, 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. 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.

De-identifying data for publication

Before publishing data, you should carefully perform a **final de-identification**. Its objective is to reduce the risk of disclosing confidential information in the published dataset. that cannot be manipulated or linked to identify any individual research participant. If you are following the steps outlined in this book, you have already removed any direct identifiers after collecting the data. At this stage, however, you should further remove all indirect identifiers, and assess the risk of statistical disclosure.[25] To the extent required to ensure reasonable privacy, potentially identifying variables must be further masked or removed.

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`[26] from IPA, `PII-scan`[27] from JPAL, and `sdcMicro`[28] from the World Bank. The `sdcMicro` tool, in particular, has a feature that allows you to assess the uniqueness of your data observations, and simple measures of the identifiability of records from that.

There will almost always be a trade-off between accuracy and privacy. For publicly disclosed data, you should favor privacy. Stripping identifying variables from a dataset may not be sufficient to protect respondent privacy, due to the risk of re-identification. One potential solution is to add noise to data, as the US Census Bureau has proposed.[29] This makes the trade-off between data accuracy and privacy explicit. But there are not, as of yet, 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,

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

[26] `https://github.com/PovertyAction/PII_detection`

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

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

[29] 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

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.

In cases where confidential data is required for analysis, we recommend embargoing sensitive or access-restricted variables when publishing the dataset. Access to the embargoed data could be granted for specific purposes, such as a computational reproducibility check required for publication, if done under careful data security protocols and approved by an IRB.

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 should be easy 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 by publication stage, your code is much less likely to be so. This is often where you need to invest time prior to releasing your replication package.

Unlike data, code usually has few legal and privacy constraints to publication. The research team owns the code in almost all cases, and code is unlikely to contain identifying information (though you must 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!)

Before releasing the code, make sure it 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, you should include a master script that allows the reviewer to run the entire project and re-create all raw outputsby changing only a single line of code: the one setting the directory path. To ensure that your code will run completely on a new computer, you must 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 code inputs and outputs are clearly identified. A new user should, for example, be able to easily find 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, so ensure 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 LaTeXdocument can support this.) Code and outputs which are not used should be removed before publication.

Releasing a replication package

Once your data and code are polished for public release, 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. One option is GitHub. Making a public GitHub repository 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. (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 the ideal platform to release reproducibility packages. It is built to version control code, and to facilitate collaboration on it. Features to look for in a platform to release such packages and that are not offered by GitHub, include: the possibility to store data and documentation as well as code, the creation of a static copy of its content, that cannot be changed or removed, and the assignment of a permanent digital object identifier (DOI) link.

Another option is the Harvard Dataverse,[30] which is designed to be a citable data repository. The Open Science Framework[31] and ResearchGate.[32] can also hold both code and data.

Any of these locations is acceptable – the main requirement is

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

[31] https://osf.io

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

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. For example, one could publish code on GitHub that points to data published on the World Bank MicroData catalog.

Emerging technologies such as the "containerization" approach of Docker or CodeOcean[33] offer to store both code and data, and also provide an online workspace in which others can execute and modify your code without having to download your tools and match your local environment when packages and other underlying software may have changed since publication.

[33] https://codeocean.com

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.[34] 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 staff may be blocked from accessing your material.

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

Bringing it all together

We hope you have enjoyed *Data for Development Impact: The DIME Analytics Resource Guide*. Our aim was to teach you to handle data more efficiently, effectively, and ethically. We laid out a complete vision of the tasks of a modern researcher, from planning a project's data governance to publishing code and data to accompany a research product. We have tried to set the text up as a resource guide so that you will always be able to return to it as your work requires you to become progressively more familiar with each of the topics included in the guide.

We started the book with a discussion of research as a public service: one that requires you to be accountable to both research participants and research consumers. We then discussed the current research environment, which necessitates cooperation with a diverse group of collaborators using modern approaches to computing technology. We outlined common research methods in impact evaluation, with an eye toward structuring data work. We discussed how to implement reproducible routines for sampling and randomization, and to analyze statistical power and use randomization inference. We discussed data collection and analysis methods, as well as tools and practices for making this work publicly accessible. Throughout, we emphasized that data work is a "social process", involving multiple team members with different roles and technical abilities. This mindset and workflow, from top to bottom, outline the tasks and responsibilities that are fundamental to doing credible research.

However, as you probably noticed, the text itself provides just enough detail to get you started: an understanding of the purpose and function of each of the core research steps. The references and resources get into the details of how you will realistically implement these tasks: from DIME Wiki pages detail specific code conventions and field procedures that our team considers best practices, to the theoretical papers that will help you figure out how to handle the unique cases you will undoubtedly encounter. We hope you will keep the book on your desk (or the PDF on your desktop) and come back to it anytime you need more information. We wish you all the best in

your work and will love to hear any input you have on ours![1]

[1] You can share your comments and suggestion on this book through https://worldbank.github.io/d4di.

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-level graduates that have joined our team tended to have very good theoretical knowledge, but have required a lot of training in practical skills. To us, this is like an architecture graduate having learned how to sketch, describe, and discuss the concepts and requirements of a new building very well – but without having the technical skills to contribute to a blueprint following professional standards that can be used and understood by other professionals. The reasons for this are a topic for another book, but in today's data-driven world, people working in quantitative development research must be proficient collaborative programmers, and that includes more than being able to compute the correct numbers.

This appendix begins with a short section containing instructions on how to access and use the code examples shared in this book. The second section contains the DIME Analytics Stata Style Guide. We believe these resources can help anyone write more understandable code, no matter how proficient they are in writing 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 increase the emphasis given to using, improving, sharing, and standardizing code style among the Stata community. Style guides are the most important tool in how you, like an architect, can 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 several 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. If you have Stata installed on your computer, then you will already 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

Whether you are new to Stata or have used it for decades, you will always run into commands that you have not seen before or whose function you do not remember. (Whether you are new or not, you should frequently revisit the most common commands – often you will learn they can do something you never realized.[1]) Every time that happens, you should always look up the help file for that command. We often encounter the conception that help files 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 help files. 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 help file for the command `isid` will open. We cannot emphasize enough how important it is that you get into the habit of reading help files. Most of us have a help file window open at all times.

[1] https://www.stata.com/manuals13/u27.pdf

Sometimes, you will encounter code that employs user-written commands, and you will not be able to read their help files until you have installed the commands. Two examples of these in our code are `randtreat` or `ieboilstart`. The most common place to distribute user-written commands for Stata is the Boston College Statistical Software Components (SSC) archive.[2] 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.

[2] https://ideas.repec.org/s/boc/bocode.html

Some commands on SSC are distributed in packages. This is the case, for example, of `ieboilstart`. That means 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 contains a command called `ieboilstart`. Stata will find `ieboilstart` in the package `ietoolkit`, so to use this command you will type `ssc install ietoolkit` in Stata instead.

We understand that it can be confusing to work with packages for first time, but this is the best way to set up your Stata installation

to benefit from other people's work that has been made publicly available. Once you get 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.[3] More commonly, there are well-recognized but non-official style guides like the JavaScript Standard Style[4] for JavaScript or Hadley Wickham's style guide for R.[5] Google, for example, maintains style guides for all languages that are used in its projects.[6]

Aesthetics is an important part of style guides, but not the main point. Neither is telling you which commands to use: there are plenty of guides to Stata's extensive functionality.[7] The important function is to allow programmers who are likely to work together to share conventions and understandings of what the code is doing. Style guides therefore help improve the quality of the code in that language that is produced by all programmers in a community. It is through a shared style that newer programmers can learn from more experienced programmers how certain coding practices are more or less error-prone. Broadly-accepted style conventions 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 would be the one adopted the most widely. What is important is that you adopt a style guide and follow it consistently across your projects.

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

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

[5] https://style.tidyverse.org/syntax.html

[6] https://github.com/google/styleguide

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

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 the way you did. In general, try to write simpler code that needs less explanation, even if you could use an elegant and complex method in less space, unless the advanced method is a widely accepted one.

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

stata-comments.do

```
TYPE 1:

/*
    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
*/

TYPE 2:

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

TYPE 3:

* 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. You can tell if a command can be abbreviated if the help file indicates an abbreviation by underlining part of the name in the syntax section at the top. Only built-in commands can be abbreviated; user-written commands cannot. (Many commands additionally allow abbreviations of options: these are always acceptable at the shortest allowed abbreviation.) 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 a accepted abbreviation exists. We do not abbreviate `local`, `global`, `save`, `merge`, `append`, or `sort`. The following is a list of 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`
`noi`	`noisily`
`cap`	`capture`
`forv`	`forvalues`
`prog`	`program`
`hist`	`histogram`

Abbreviating variables

Never abbreviate variable names. Instead, write them out completely. Your code may change if a variable is later introduced that has a name exactly as in the abbreviation. `ieboilstart` executes the command `set varabbrev off` by default, and will therefore break any code using variable abbreviations.

Using wildcards and lists in Stata for variable lists (`*`, `?`, and `-`) is also discouraged, because the functionality of the code may change

if the dataset is changed or even simply reordered. If you intend explicitly to capture all variables of a certain type, prefer `unab` or `lookfor` to build that list in a local macro, which can then be checked to have the right variables in the right order.

Writing loops

In Stata examples and other code languages, it is common for the name of the local generated by `foreach` or `forvalues` to be 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. Even counters should be explicitly named. This makes code much more readable, particularly in nested loops.

stata-loops.do

```
BAD:

* Loop over crops
foreach i in potato cassava maize {
   * do something to `i'
}

GOOD:

* Loop over crops
foreach crop in potato cassava maize {
   * do something to `crop'
}

GOOD:

* Loop over crops
local crops potato cassava maize
  foreach crop of local crops {
        * Loop over plot number
        forvalues plot_num = 1/10 {
          * do something to `crop' in `plot_num'
        } // End plot loop
  } // End crop loop
```

Using whitespace

In Stata, adding one or many spaces does not make a difference to code execution, and this can be used to make the code much more

readable. 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. The same principles apply to coding. In the example below the exact same code is written twice, but in the better 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 it much quicker to read, and small typos stand out more, making them easier to spot.

stata-whitespace-columns.do

```
ACCEPTABLE:

    * Create dummy for being employed
    gen employed = 1
    replace employed = 0 if (_merge == 2)
    lab var employed "Person exists in employment data"
    lab def yesno 1 "Yes" 0 "No"
    lab val employed yesno

BETTER:

    * Create dummy for being employed
    gen      employed = 1
    replace  employed = 0 if (_merge == 2)
    lab var  employed "Person exists in employment data"
    lab def           yesno 1 "Yes" 0 "No"
    lab val  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. 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

```
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
    }

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 (`~`). Always use explicit truth checks (`if `value'==1`) rather than implicits (`if `value'`). Always use the `missing(`var')` function instead of arguments like (`if `var'<=.`). Always consider whether missing values will affect the evaluation of conditional expressions.

stata-conditional-expressions1.do

```
GOOD:

    replace gender_string = "Woman" if (gender == 1)
    replace gender_string = "Man"   if ((gender != 1) & !missing(gender))

BAD:

    replace gender_string = "Woman" if gender == 1
    replace gender_string = "Man"   if (gender ~= 1)
```

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, which makes your code more readable. It is also less error-prone and easier to update if you want to change the condition.

stata-conditional-expressions2.do

```
GOOD:

    if (`sampleSize' <= 100) {
        * do something
    }
    else {
        * do something else
    }

BAD:

    if (`sampleSize' <= 100) {
        * do something
    }
    if (`sampleSize' > 100) {
        * do something else
    }
```

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**. All macros should be defined using the = operator. Never abbreviate the commands for **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 curly brackets around their name (`${}`); otherwise, they can cause readability issues when the endpoint of the macro name is unclear.

You should use descriptive names for all macros (up to 32 characters; preferably fewer). 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. Simple prefixes are useful and encouraged such as `this_estimate` or `current_var`, or, using `camelCase`, `lastValue`, `allValues`, or `nValues`. Nested locals (``` ``value'' ```) are also possible for a variety of reasons when looping, and should be indicated in comments. If you need a macro to hold a literal macro name, it can be done using the backslash escape character; this causes the stored macro to be evaluated at the usage of the macro rather than at its creation. This functionality should be used sparingly and commented extensively.

stata-macros.do

```
GOOD:

    global     myGlobal = "A string global"
    display "${myGlobal}"

BAD:

    global my_Global = "A string global" // Do not mix naming styles
    display "$myGlobal"                  // Always use ${} for globals
```

Writing file paths

All file paths should always be enclosed in double quotes, and should always use forward slashes for folder hierarchies (`/`). File names should be written in lower case with dashes (`my-file.dta`). Mac and Linux computers cannot read file paths with backslashes, and backslashes cannot easily be removed with find-and-replace because the character has other functional uses in code. File paths should always include the file extension (`.dta`, `.do`, `.csv`, etc.). Omitting the extension causes ambiguity if another file with the same name is created (even if there is a default file type).

File paths should also be absolute and dynamic. **Absolute** means that all file paths start at the root folder of the computer, often `C:/` on a PC or `/Users/` on a Mac. This ensures that you always get the correct file in the correct folder. Do not use `cd` unless there is a function that requires it. When using `cd`, it is easy to overwrite a file

in another project folder. Many Stata functions use `cd` and therefore the current directory may change without warning. Relative file paths are common in many other programming languages, but there they are always relative to the location of the file running the code. Stata does not provide this functionality.

Dynamic file paths use global macros for the location of the root folder. These globals should be set in a central master do-file. This makes it possible to write file paths that work very similarly to relative paths. It also achieves the functionality that setting `cd` is often intended to: executing the code on a new system only requires updating file path globals in one location. If global names are unique, there is no risk that files are saved in the incorrect project folder. You can create multiple folder globals as needed and this is encouraged.

stata-filepaths.do

```
GOOD:

    global myDocs    = "C:/Users/username/Documents"
    global myProject = "${myDocs}/MyProject"
    use "${myProject}/my-dataset.dta" , clear

BAD:

    RELATIVE PATHS:
    cd "C:/Users/username/Documents/MyProject"
    use MyDataset.dta

    STATIC PATHS:
    use "C:/Users/username/Documents/MyProject/MyDataset.dta"
```

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. Stata's do-file editor and other code editors provide a visible guide line. Around that length, start a new line using `///`. Using `///` breaks the line in the code editor, while telling Stata that the same line of code continues on the next line. The `///` breaks do not need to be horizontally aligned in code, although you may prefer to if they have comments that read better aligned, since indentations should reflect that the command continues to a new line. Break lines where it makes functional sense. You can write comments after `///` just as with `//`, and that is usually a good thing. The `#delimit` command should only be used for advanced function programming and is officially discouraged in analytical code.[8] Never, for any reason, use

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

`/* */` to wrap a line: it is distracting and difficult to follow compared to the use of those characters to write regular comments. Line breaks and indentations may be used to highlight the placement of the **option comma** or other functional syntax in Stata commands.

stata-linebreak.do

```
GOOD:
    graph hbar invil      /// Proportion in village
         if (priv == 1)   /// Private facilities only
       , over(statename, sort(1) descending)      /// Order states by values
         blabel(bar, format(%9.0f))               /// Label the bars
         ylab(0 "0%" 25 "25%" 50 "50%" 75 "75%" 100 "100%") ///
         ytit("Share of private primary care visits made in own village")

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

UGLY:
    graph hbar /*
*/        invil if (priv == 1)
```

Using boilerplate code

Boilerplate code is a few lines of code that always come at the top of the code file, and its purpose is to harmonize settings across users running the same code to the greatest degree possible. There is no way in Stata to guarantee that any two installations 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. We have developed the `ieboilstart` command to implement many commonly-used boilerplate settings that are optimized given your installation of Stata. It requires two lines of code to execute the `version` setting, which avoids differences in results due to different versions of Stata. Among other things, it turns the `more` flag off so code never hangs while waiting to display more output; it turns `varabbrev` off so abbrevated variable names are rejected; and it maximizes the allowed memory usage and matrix size so that code is not rejected on other machines for violating system limits. (For example, Stata/SE and Stata/IC, allow for different maximum numbers of variables, and the same happens with Stata 14 and Stata 15, so it may not be able to run code written in one of these version using another.) Finally, it clears all stored information in Stata memory, such as non-installed programs and globals, getting as close

as possible to opening Stata fresh.

stata-boilerplate.do

```
GOOD:

    ieboilstart, version(13.1)
    `r(version)'

OK:

    set more off , perm
    clear all
    set maxvar 10000
    version 13.1
```

Saving data

There are good practices that should be followed before saving any dataset. These are to `sort` and `order` the dataset, dropping intermediate variables that are not needed, and compressing the dataset to save disk space and network bandwidth.

If there is a unique ID variable or a set of ID variables, the code should test that they are uniqueally and fully identifying the dataset.[9] ID variables are also perfect variables to sort on, and to `order` first in the dataset.

The command `compress` makes the dataset 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.

[9] https://dimewiki.worldbank.org/ID_Variable_Properties

stata-before-saving.do

```
* If the dataset has ID variables, test if they 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

    save    "${myProject}/myDataFile.dta" , replace // The folder global is set in master do-file
    saveold "${myProject}/myDataFile-13.dta" , replace v(13) // For others
```

Miscellaneous notes

Write multiple graphs as `tw (xx)(xx)(xx)`, not `tw xx||xx||xx`.

In simple expressions, put spaces around each binary operator except `^`. Therefore write `gen z = x + y` and `x^2`.

When order of operations applies, you may adjust spacing and parentheses: write `hours + (minutes/60) + (seconds/3600)`, not `hours + minutes / 60 + seconds / 3600`. For long expressions, `+` and `-` operators should start the new line, but `*` and `/` should be used inline. For example:

```
gen newvar = x ///
 - (y/2) ///
 + a * (b - c)
```

Make sure your code doesn't print very much to the results window as this is slow. This can be accomplished by using `run file.do` rather than `do file.do`. Interactive commands like `sum` or `tab` should be used sparingly in do-files, unless they are for the purpose of getting `r()`-statistics. In that case, consider using the `qui` prefix to prevent printing output. It is also faster to get outputs from commands like `reg` using the `qui` prefix.

Bibliography

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

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.

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

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.

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.

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.

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.

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

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

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

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.

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.

Callen, M. (2015). Catastrophes and time preference: Evidence from the indian ocean earthquake. *Journal of Economic Behavior & Organization*, 118:199–214.

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.

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

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

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

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

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

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.

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

Glewwe, P. and Grosh, M. E. (2000). *Designing household survey questionnaires for developing countries: lessons from 15 years of the living standards measurement study*. World Bank.

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

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

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

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

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

Imbens, G. W., Rubin, D. B., and Sacerdote, B. I. (2001). Estimating the effect of unearned income on labor earnings, savings, and consumption: Evidence from a survey of lottery players. *American economic review*, 91(4):778–794.

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

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

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

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

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

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.

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

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.

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

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.

Schwarzer, G., Carpenter, J. R., and Rücker, G. (2015). Small-study effects in meta-analysis. In *Meta-analysis with R*, pages 107–141. Springer.

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

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

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

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.

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.

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

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

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

LaTeX, 104
`DataWork` folder, 33
`iefolder`, 33
`ietoolkit`, 33
LaTeX, 34, 41

anonymization, 110
attrition, 74
average treatment effect, 44

backup, 28
binary files, 34

causal inference, 45
code organization, 36
code review, 38
collaboration tools, 30
contamination, 74
control group, 45
counterfactual, 44
credibility, 17

data analysis, 98
data collection, 21
data organization, 88
data ownership, 22
data storage, 23
data transfer, 23
data visualization, 100
de-identification, 21, 25
difference-in-differences, 48
Documentation, 94
Dropbox, 28

dynamic documents, 34

email, 29
encryption, 23, 28, 82

file paths, 28
file sharing, 29
file syncing, 29

geodata, 21
GitHub, 20

human subjects, 24

identification, 43
iecodebook, 92
iefieldkit, 92
Institutional Review Board, 22

Markdown, 33
master do-file, 37
matching, 52

Open Science Framework, 20

password protection, 28
personally-identifying
    information, 21, 90
plaintext, 34
Pre-analysis plan, 94, 95
pre-analysis plan, 20
pre-analysis plans, 19
pre-registration, 19, 20

primary data, 21
privacy, 21
project documentation, 19

questionnaire design, 73

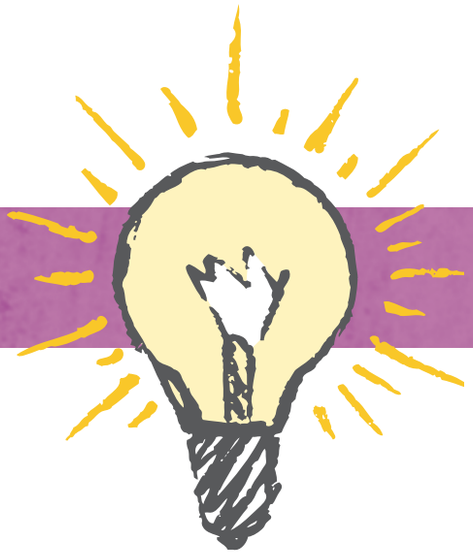
randomized control trials, 45
Registered Reports, 19
regression discontinuity, 50
reproducibility, 17
Research design, 94
running variable, 50

sampling, 59
server storage, 29
software environments, 31
software versions, 31
statistical disclosure, 110
synthetic controls, 53

task management, 20
transparency, 17
treatment effect, 43

version control, 29

WhatsApp, 29



WORLD BANK GROUP

i2i DIME TRANSFORM DEVELOPMENT

DIME analytics

15173-Analytics_coverFINAL.indd 2 5/17/19 2:45 PM