KRISTOFFER BJÄRKEFUR
LUÍZA CARDOSO DE ANDRADE
BENJAMIN DANIELS
MARIA JONES

DEVELOPMENT RESEARCH IN PRACTICE:

THE DIME ANALYTICS DATA HANDBOOK

DIME ANALYTICS

Kristoffer Bjärkefur
Luíza Cardoso de Andrade
Benjamin Daniels
Maria Jones

PUBLISHED BY DIME ANALYTICS

https://worldbank.github.com/d4di

First printing, July 2020

Compiled from: https://github.com/worldbank/dime-data-handbook/commit/

Notes on this edition

This is a draft peer review edition of *Development Research in Practice: The DIME Analytics Data Handbook*. 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/dime-data-handbook`. You can get a PDF copy at: `https://worldbank.github.com/dime-data-handbook`. 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/dime-data-handbook/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: Development research in practice

17 Chapter 1: Reproducibility, transparency, and credibility

27 Chapter 2: Setting the stage for collaboration

43 Chapter 3: Establishing a measurement framework

59 Chapter 4: Acquiring development data

75 Chapter 5: Cleaning and processing research data

89 Chapter 6: Analyzing research data

103 Chapter 7: Publishing research outputs

117 Bringing it all together

119 Appendix: The DIME Analytics Stata Style Guide

133 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: Development research in practice

Welcome to *Development Research in Practice*. 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 US$180

[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

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. 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.

Development Research in Practice compiles the ideas, best practices and software tools that the DIME Analytics team has developed while supporting DIME's global impact evaluation portfolio. Each chapter in this book 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.

We will not always give a lot of highly specific implementation details in this text, but will often point you to where they can be found on the **DIME Wiki**.[4] The DIME Wiki is one of DIME Analytics' flagship products, a free online collection of our resources and best practices.[5] This book complements the DIME Wiki by providing a structured narrative of the data workflow for a typical research project. The Wiki, by contrast, provides unstructured but detailed information on how to complete each task, and links to further practical resources.

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

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

Adopting reproducible practices through code

We assume throughout 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, and think of code and data as research outputs, just as manuscripts and briefs are.

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, we 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, as an appendix, the **DIME Analytics Stata Style Guide** that we use in our work, which provides standards for coding in Stata 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 one 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 this 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, in that 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 and file organization your code lives in: good structure means that it is easy to find individual pieces of code that correspond to specific tasks and outputs. 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?

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.

[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 reproducibility, transparency, and credibility. In Chapter 1, we outline a set of practices that help to ensure that research consumers can be confident in the conclusions reached, and research work can be assumed and verified to be reliable. Chapter 2 will teach you to structure your data work to be organized and collaborative, while ensuring the privacy and security of research participants. It discusses the importance of planning these data tools and structures 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 establishing a measurement framework, focusing specifically on how to translate research design to master datasets, create documentation of the structure of your data, and how to implement both simple and complex randomized designs reproducibly.

Chapter 4 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 5 teaches workflows for data processing. It details how to construct "tidy" data at the appropriate units of analysis, how to ensure uniquely identified datasets, and how to routinely incorporate data quality checks into the workflow. It also provides guidance on de-identification and cleaning of personally-identified data, focusing on how to understand and structure data so that it is ready for indicator construction and analytical work. Chapter 6 discusses data analysis. It begins with data construction, or the creation of new variables from the raw information collected or obtained in the field. It also introduces core principles for writing analytical code and creating, exporting, and storing research outputs such as figures and tables reproducibily with dynamic documents. In Chapter 7, we turn to publication. This chapter discusses how to effectively collaborate on technical writing, how and why to publish data, and guidelines for preparing functional and informative reproducibility 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: Reproducibility, transparency, and credibility

Policy decisions are made every day using the results of development briefs and studies, and these have wide-reaching effects on the lives of millions. As the range of policy questions asked by researchers grows, so too does the scrutiny under which research methods and results are placed. Three major components make up this scrutiny: credibility, transparency, and reproducibility. These three components contribute to one simple idea: research should be high quality and well-documented. It should be able to be easily examined and recreated by readers and policymakers who will make choices based on that evidence. In this framework, is useful to think of research as a public service that requires researchers as a group to be accountable to their methods. This means acting to collectively protect the credibility of development research by following modern practices for research planning, transparency, and reproducibility.

Across the social sciences, the open science movement has been fueled by concerns about the proliferation 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 improvements in the way that code and data are handled as part of open research. As a community, having common standards and practices for creating and sharing materials, code, and data with others will improve the value of the work we do. In this chapter, we outline principles and practices that help to ensure research consumers can be confident in the conclusions reached. We share our experience implementing best practices in open science research at DIME, and focus on practical guidance tested on projects across our portfolio.

Developing a credible research project

The evidentiary value of research is traditionally a function of design choices.[1] Was the research design sufficiently powered through its sampling and randomization? Were the key research outcomes pre-specified or chosen ex-post? How sensitive were the results to changes in specifications or definitions? Reproducible and transparent methods are key to maintaining credibility in these choices and avoiding serious errors.[2] This is especially relevant for research that relies on original or novel data sources, from innovative big data sources to survey datasets in unique contexts. One frequent target for critics of such research[3] is the fact that most researchers have a lot of leeway in selecting the projects, results, or outcomes they focus on *after* already having had the experience of implementing a project or collecting data in the field, which increases the likelihood of finding

[1] 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

[2] Christensen, G., Freese, J., and Miguel, E. (2019). *Transparent and reproducible social science research: How to do open science*. University of California Press

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

"false positive" results that are not true outside carefully-selected data. Development researchers should take these concerns seriously. Such flexibility can pose a serious issue for the quality of evidence overall, particularly if researchers believe that certain types of results are substantially better for their careers or their publication chances.

This section presents three popular methods for researchers to commit to particular research questions or methods, and to avoid potential criticisms of cherry-picking results for publication. Each of these methods involves putting all or part of a research study into a formal document or registry, before carrying it out if possible. Registering studies, for example, provides notice that a study has been done; *pre*-registering them provides notice that a study was attempted (even if never completed). Pre-analysis plans extend this concept to a more formal commitment to use specific methods on particular questions. Writing and releasing a pre-analysis plan in advance of working with data is therefore often used to protect the credibility of approaches that have a high likelihood of returning false results if decided ex post. Finally, registered reports allow researchers to approach research planning itself as a process at the level of a full peer review. Registered reports enable close scrutiny of a project design, a feedback and improvement process, and a commitment from a publisher to publish specific results that are agreed to be studied with specific methods.

Registering research studies

Registration of research studies is a common practice intended to ensure that a complete record of research inquiry is easily available.[4] Registering research studies ensures that future scholars can quickly find out what work has been carried out on a given question, even if some or all of the work done never results in formal publication. Registration of studies is increasingly required by publishers and can be done very quickly either before, during, or after the study with only basic information about the study purpose. Some common registries are operated by the **AEA**,[5] **3ie**,[6] **eGAP**,[7] and **OSF**,[8]. They all have different target audiences and features, so select one that is appropriate to your work.

Pre-registering studies *before they begin work* is a further extension of this principle.[9] Registration of a study before it goes to field, particularly when specific hypotheses are included in the registration, provides a simple and low-effort way for researchers to conclusively demonstrate that a particular line of inquiry was not generated by the process of data collection or analysis itself. Pre-registrations need not provide exhaustive details about how a particular hypothesis

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

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

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

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

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

[9] Nosek, B. A., Ebersole, C. R., DeHaven, A. C., and Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606

will be approached; only that it will be.[10] This can be highly valuable and requires very little additional investment. As a result, the DIME team requires pre-registration of all studies in a public database with at least some primary hypotheses prespecified, prior to providing funding for impact evaluation research.

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

Writing pre-analysis plans

If a researcher has a large amount of flexibility to define how they approach a particular hypothesis, study registration may not be sufficient to avoid the criticism of "hypothesizing after the results are known", or HARKing.[11] In other cases, a researcher will know that they will still have a large amount of flexibility to define exactly how they approach a particular hypothesis. This could include a diverse set of measures of an abstract concept; it could include future choices about sample inclusion or exclusion; or it could include decisions about how to construct derived indicators. When the researcher is collecting a large amount of information and has leverage over even a moderate number of these options, it is almost guaranteed that they can come up with any result they like.[12]

[11] Kerr, N. L. (1998). Harking: Hypothesizing after the results are known. *Personality and Social Psychology Review*, 2(3):196–217

[12] Gelman, A. and Loken, E. (2013). The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University*

Pre-analysis plans can be used to assuage these concerns for evaluations by fully specifying some set of analyses intended to be conducted.[13] In particular, such a plan should be written up in detail for areas that are known to provide a large amount of leeway for researchers to make later decisions, particularly for things like interaction effects or subgroup analysis. Even when used, pre-analysis plans shoud not be viewed as binding the hands of the researcher.[14] Depending on what is known about the study at the time of writing, pre-analysis plans can vary widely in the amount of detail they should include.[15] The core function of a plan is to carefully and explicitly describe one or more specific data-driven inquiries, as specific formulations are ofen very hard to justify in retrospect with data or projects that potentially provide many avenues to approach a single theoretical question. 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 the details of a particular inquiry makes its results immune to a wide range of criticisms of specification searching or multiple testing.[16]

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

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

[15] https://blogs.worldbank.org/impactevaluations/pre-analysis-plans-and-registered-reports-what-new-opinion-piece-does-and-doesnt

[16] Duflo, E., Banerjee, A., Finkelstein, A., Katz, L. F., Olken, B. A., and Sautmann, A. (2020). In praise of moderation: Suggestions for the scope and use of pre-analysis plans for rcts in economics. Technical report, National Bureau of Economic Research

Publishing registered reports

Registered reports[17] take the process of pre-planning a complex research design to the level of a formal publication. In a registered report, a journal or other publisher will accept a particular format of study description for publication, typically then guaranteeing the ac-

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

ceptance of a later publication that carries out the analysis described in the registered report. While far stricter and more complex to carry out than ordinary study registration or pre-analysis planning, the registered report has the added benefit of soliciting peer review and expert feedback on the design and structure of the proposed study.

This process is in part meant to combat the "file-drawer problem",[18] and ensure that researchers are transparent in the sense that all promised results obtained from registered-report studies are actually published. This approach has the advantage of pre-specifying in great detail a complete research and analytical design, and securing a commitment for publication regardless of the outcome. This may be of special interest for researchers studying events or programs where either there is a substantial risk that they would either not be able to publish a null or negative result, or where they may wish to avoid any pressure toward finding a particular result when the program or event is the subject of substantial social or political pressures. As with pre-registration and pre-analysis, nothing in a registered report should be understood to prevent a researcher from pursuing additional avenures of inquiry once the study is complete, either in the same or separate research outputs.

[18] 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

Documenting and cataloguing transparent research

Clearly documenting research work is the next step in allowing others to evaluate exactly what was done and measured to obtain a particular result. Many development research projects are purpose-built to address specific questions, and often use unique data, novel methods, or small samples. These approaches can yield new insights into essential academic questions, but need to be transparently documented so they can be reviewed or replicated by others in the future.[19] Contrasted with disciplines where data is more standardized or where research is more oriented around secondary data, it is often the case that the exact data used in a development project has never been observed by anyone else in the past and may not be able to be re-collected by others in the future. Whether or not this is the case, transparent documentation methods help ensure that data was collected and handled appropriately and that studies and interventions were implemented correctly. As with study registrations, project and data documentation should be released on external archival repositories so they can always be accessed and verified.

[19] 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

Documenting data acquisition and analysis

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. 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. Some work better for field records such as implementation decisions, research design, and survey development; others work better for recording data work and code development. The **Open Science Framework**[20] provides one such solution, with integrated file storage, version histories, and collaborative wiki pages. **GitHub**[21] provides a transparent documentation system,[22] 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 documentation service, because communications are rarely well-ordered, can be easily deleted, and are not available for future team members.)

[20] https://osf.io

[21] https://github.com

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

Cataloging and archiving data acquisition

Data and data collection methods should be fully cataloged, archived, and documented, whether you are collecting data yourself or recieving it from an outside partner. In some cases this is as simple as uploading a PDF survey instrument or data catalog and a coded survey definition form or codebook to an archive. In other cases this will be more complex. Proper documentation of data collection will often require a detailed description of the overall sampling procedure. For example, settings with many overlapping strata, treatment arms, excluded observations, or resampling protocols might require extensive additional field work documentation. This documentation should be continuously updated and kept with the other study materials; it is often necessary to collate these materials for an appendix for any publication in any case.

The raw dataset that comes in from the field should also be immediately archived and cataloged. Some project funders, such as government agencies, NGOs, and nonprofits provide specific respositories in which they require the deposit of data they funded,[23] and you should take advantage of these when possible. If this is not provided, you must be aware of privacy issues with directly identifying data and questions of data ownership before uploading raw data to any third-party server, whether public or not; this is a legal question for your home organization. This type of data depositing or archiving is different from publishing or releasing the data: data at this stage may still need to be embargoed or have other, potentially permanent, access restrictions. Similarly, data catalogs (such as the World Bank Data Catalog[24]) do not actually make data accessible: they simply create a record of its existence and provide instructions on how access would be obtained.

[23] For example, urlhttps://data.usaid.gov

[24] urlhttps://datacatalog.worldbank.org

Archiving and cataloging data does not resolve the need to create an authoritative, citable, *published* data set in a publicly accessible first- or third-party archive, even if access restrictions are still required. For more on the steps required to prepare and publish a de-identified dataset, you can refer to Chapter 6 and Chapter 7 of this book. You should use a tool such as a Dataverse or the OSF to upload and store de-identified and publishable versions of your data, after archiving the complete data in an institution-approved location. Data publication should create a data citation and a digital object identifier (DOI), or some other persistent index that you can use in your future work to unambiguously indicate the location of your data. This data publication should also include the methodological documentation as well as complete human-readable codebooks for all the variables there.

Preparing for reproducible analysis

Development research is rapidly moving in the direction of requiring strict adherence to specific reproducibility guidelines.[25] 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.[26] Common research standards from journals and funders feature both ex ante (or "regulation") and ex post (or "verification") policies.[27] Ex ante policies require that authors provide replication materials before publication which are then reviewed by the journal for completeness. Ex post policies require that authors make certain materials available to the public, but their completeness is not a precondition for publication. Other journals have adopted "guidance" policies that offer checklists for reporting on whether and how various practices were implemented, without specifically requiring any.[28] Documentation on data processing and additional hypotheses tested will be expected in the supplemental materials to any publication.

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

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

[27] 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

[28] 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

At DIME, all research outputs are expected to be reproducible. Before releasing a working paper, a research team is required to submit to the DIME Analytics team a fully reproducible analysis package with de-identified data, which is then independently verified to produce the same results as appear in the working paper, as well as ensuring sufficient documentation to operate. (The team has never received a package that worked perfectly on the first try.) Additionally, the team organizes frequent peer code review of works in progress, and our general recommendation is to ensure that projects are *always* externally reproducible instead of waiting until the final stages to prepare this material. Once completed, the team receives a completed reproducibility certificate that also lists any publicly available materials to accompany the package, which is encouraged to be included as an appendix to the publication.

Reproducible research is a public good

Transparent research exposes not only the code, but all research processes involved in developing the analytical approach.[29] 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[30] is shared, this makes it easy for the reader to understand the analysis fully. 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

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

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

for themselves, because it requires methodical organization that is labor-saving over the complete course of a project.

Making your research reproducible is also a public good.[31] 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 corresponding analytical dataset should always be made openly accessible to the greatest legal and ethical extent that it can be.[32]

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

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

Completing a reproducibility package

Can another researcher reuse the same code on the same data and get the exact same results as in your published paper?[33] This is a standard known as **computational reproducibility**, and it is an increasingly common requirement for publication.[34] 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. As we will discuss in Chapter 5, code that is well-organized into a master script, and written to be easily run by others, makes this task simpler.

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

[34] 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 confidential 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.[35] 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.[36]

A reproducibility package should include the complete materials needed to exactly re-create your final analysis, and be accessible and well-documented so that others can identify and adjust potential decision points that they are interested in. They should be able to

[35] 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

easily identify what data was used for each output, as well as be able to reproduce each analytical component independently of all others. A well-organized reproducibility package usually takes the form of a complete directory structure, including documentation and a master script, that leads the reader through the process and rationale for the code behind each of the outputs when considered in combination with the corresponding publication.

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. Every investment you make in documentation and transparency up front protects your project down the line, particularly as these standards continue to tighten.

Chapter 2: Setting the stage for collaboration

Preparation for collaborative data work begins long before you acquire any data, and involves planning both software tools and collaboration platforms and processes for your team. In order to be prepared to do effective data work in a team environment, you need to structure your workflow in advance. This means knowing what types of data you'll acquire, whether the data will require special handling due to size or privacy considerations, which datasets and outputs you will need at the end of the process, and how all data files and versions will stay organized throughout. It's important to plan data workflows in advance because changing software or protocols halfway through a project can be costly and time-consuming. Seemingly small decisions such as sharing services, folder structures, and filenames can be extremely painful to alter down the line in any project.

This chapter will guide you in setting up an effective environment for collaborative data work, structuring your data work to be well-organized and clearly documented, and setting up processes to handle confidential data securely. The first section outlines hows to set up your working environment to effectively collaborate on technical tasks with others, and how to document tasks and decisions. The second section discusses how to organize your code and data so that others will be able to understand and interact with it easily. The third section provides guidelines for ensuring privacy and security when working with confidential data.

Preparing a collaborative work environment

This section introduces the core concepts and tools for organizing 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 form a collaborative workflow that will greatly accelerate your team's ability to get tasks done on all your projects.

Teams often develop workflows in an ad hoc fashion, 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. For example, documentation, naming schema, folder and output organization, coding, managing revisions to files, and reviewing each other's work. These tasks are common to almost

every project, and solutions translate well between projects. Therefore, there are large efficiency gains to thinking in advance about the best way to do these tasks, instead of throwing together a solution when the task arises. This section outlines the main points to discuss within the team, and suggests best practice solutions for these tasks.

Setting up your computer

First things first: almost all your data work will be done on your computer, so make sure it's set up for success. The operating system should be fully updated, it should be in good working order, and you should have a **password-protected** login.

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 (for example, any type of code 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 that 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

Good collaboration tools are task-oriented systems that allow 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 status. They are web-based so that everyone on your team can access them simultaneously and have live discussions about tasks and processes. Such systems link communication to specific tasks so that related decisions are permanently recorded and easy to find in the future when questions about that task come up. Choosing the right tool for your team's needs is essential to designing an effective workflow. What is important is that your team chooses a system and commits to using it, so that decisions, discussions, and tasks are easily reviewable long after they are completed.

Some popular and free collaboration tools that meet these criteria are GitHub project boards, GitHub issues and Dropbox Paper. Any specific list of software will quickly be outdated; we mention these as examples that have worked for our team. Different collaboration tools can be used 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, assignments, and deadlines, and is very intuitive for people with non-technical backgrounds. Therefore, it is a useful tool for managing non-code-related tasks.

Choosing software

Choosing the right software environments can make your work significantly easier. It may be difficult or costly to switch halfway through a project, so think ahead about the various software your team will use. Take into account the technical abilities of team members, how important it is to access files offline constantly, the type of data you will need to access, and the level of security required. Big datasets require additional infrastructure and may overburden the tools commonly 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 plan in advance and understand how the chosen tools will interact with your workflows.

Ultimately, the goal is 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 costly process of migrating an ongoing project into a new code enviroment.

Code environment should be constant down to the software level: the specific versions of software and the individual packages you use should be referenced or maintained so that they can be reproduced going forward, even if different releases contain changes that would break your code or change your results. DIME Analytics developed the command `ieboilstart` in the `ietoolkit` package to support Stata version stability.[5]

[5] 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 one of the two most popular statistical software packages: R or Stata. (If you are using another language, like Python, many of the same principles apply but the specifics will be different.) Most of your code work will be done in a code editor. This does not need to be the same program the code runs in; using an external editor can be preferable since your editor will not crash if your code does, and the editor may offer additional features aimed at writing code well. If you are working in R, **RStudio** is the typical choice.[6] For Stata, the built-in do-file editor is the most widely adopted code editor, but we recommend using an external editor such as **Atom**[7] or **Sublime**[8]. These editors can be configured to run Stata code externally and offer great accessibility and quality features. For example, they can access an entire directory – rather than a single file – which gives you directory-level views and file management actions, such as folder management, Git integration, and simultaneous work with other types of files, without leaving the editor.

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

[7] https://atom.io

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

Organizing code and data

We assume you are going to do your analytical work through code, and that you want all your processes to be documented and replicable. Though it is possible to interact with 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 provides basic tips on managing the folder that stores your project's data work.

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

Once you start a research project, 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. You should agree with your team on a specific directory structure, and set it up at the beginning of the project. You should also agree on a file naming convention.[9] This will help you to easily find project files and ensure that all team members can easily run the same code.

To support consistent folder organization at DIME, DIME Analytics created `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 unique needs, we have tried to make the structure easy to adapt. Having a universally standardized folder structure across the entire portfolio of projects means that everyone can easily move between projects without having to reorient on file and folder organization.

If you do not already have a standard file structure across projects, `iefolder` is an easy template to start from. The command creates a `DataWork` folder at the project level, and within that folder, creates standardized directory structures for each data source or survey round. Within each subdirectory, `iefolder` creates folders for raw encrypted data, raw deidentified data, cleaned data, final data, outputs, and documentation. It creates a parallel folder structure for the code files that move the data through this progression, and for the final analytical outputs. The `ietoolkit` package also includes

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

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

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

the `iegitaddmd` command, which can place `README.md` placeholder files in your folders so that your folder structure can be shared using Git.[12] 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.

[12] Git only tracks files, so empty folders – which most folders are in the beginning of a project – are ignored if placeholder files are not used, leading to only parts of the folder structure being shared across the team.

The `DataWork` folder may be created either inside an existing project folder, or it may be created as a separate root folder. We advise keeping project management materials (such as contracts, Terms of Reference, briefs and other administrative or management work) separate from the datawork folder structure. It is useful to maintain project management materials in a synced location like Dropbox, whereas the code folder should 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. Combining these two types of collaboration tools almost always creates undesired functionalities.)

Setting up a version control system

We recommend using a **version control system** to maintain control of file history and functionality. A good version control system tracks who edited each file and when, allows you to revert to previous versions, and 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 version-control of all data work. Git manages changes to all **plaintext** files, this includes all code files, most raw outputs, and written outputs that use plaintext tools such as LaTeX and dynamic documents. Git tracks all the changes you make to each plaintext file, 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 file, so you don't risk breaking code for other team members as you try something new.

Documenting and organizing code

Good code is written in a way that is easily understood and run by others. 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.

To be readable, code must be well-documented. 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.

Two types of comments should be included in the script itself. 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 use to communicate with your team) will make it easier for everyone to read and understand the code's purpose. It can also help you organize your own work and ensure you are following logical steps. 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 files should be stored in an easy-to-find location and named in a meaningful way. Breaking your code into independently read-

able "chunks" is good practice for code organization. You should write each functional element as a chunk that can run completely on its own. This ensures that each code component is independent; it does not rely on a complex program state created by other code chunks that are not obvious from the immediate context. One way to do this is by creating sections in your script to identify where a specific task is completed. For example, if you want to find the line in your code where the directory is set, you can go straight to `PART 2: Prepare folder paths and define programs`, instead of reading line by line through the entire code. RStudio 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 (see line 24 of the example do-file), though they're just there to make the reading easier and don't have functionality. Since an index is not automated, create this manually in the code header by copying and pasting section titles (see 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, we recommend 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.[14] 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.

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

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 sub-master do-files and add to them whenever new subfolders are created in the `DataWork` folder.

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
--------------------------------------------------------------------------- */
```

Preparing to handle confidential 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)**.[15] PII is information which can, without any transformation or linkage, be used to identify individual people, households, villages, or firms (or other units) that were part of a data collection. 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".[16] If you interact with European institutions or persons, you will also become familiar with the General Data Protection Regulation (GDPR),[17] a set of regulations governing **data ownership** and privacy standards. No matter who you are or what exact legal requirements you face, the core principles and practices you need to consider will always be similar.

[15] **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

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

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

Obtaining ethical approval and consent

Most of the field research done in development involves human subjects.[18] As a researcher, you are asking people to trust you with personal information about themselves: where they live, how rich they are, whether they have committed or been victims of crimes, their names, their national identity numbers, and all sorts of other data. PII data carries strict expectations about data storage and handling, and it is the responsibility of the research team to satisfy these expectations.[19] Your donor or employer will most likely require you to hold a certification from a source such as Protecting Human Research Participants[20] or the CITI Program.[21]

For almost all such data collection and research activities, you will be required to complete some form of **Institutional Review Board (IRB)** process.[22] Most commonly this consists of a formal application for approval of a specific protocol for consent, data collection, and data handling.[23] 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

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

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

[20] https://phrptraining.com

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

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

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

be affected by the research design. Some jurisdictions (especially those 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.[24] 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.

[24] 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.

Ensuring privacy and security in research data

In order to safeguard PII and protect respondent privacy, you must set up a secure data system from the outset of a project.[25] There are several components to this. First, you need a system for managing strong and unique passwords for all accounts – including personal accounts like computer logins and email. There are several services that create and store these passwords for you, and some provide utilities for securely sharing passwords with others. Second, all machines should have **hard disk encryption** enabled.[26] Disk encryption is available on many modern operating systems; it prevents your files from ever being accessed without first entering the system password. As with all critical passwords, your system password should be strong, memorable, and backed up in a separate secure location.

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

[26] **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

Third, you must encrypt datasets that include confidential information during data collection, storage, and transfer. 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. There are many encryption options available, from enterprise-grade solutions to free software. DIME uses Veracrypt for file encryption, our protocols are available on GitHub[27]

[27] https://github.com/worldbank/dime-standards/blob/master/dime-research-standards/pillar-4-data-security/data-security-resources/veracrypt-guidelines.md

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. Some examples of PII variables include names, addresses, and geolocations, email addresses, phone numbers, and bank accounts or other financial details. If you are working in a context or population 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 disclose their identify, even though those variables would not be considered PII in general. There is no one-size-fits-all solution to determine what is PII, research teams have to use careful judgment in each case to avoid statistical disclosure.[28] It is important to keep in mind that data privacy principles apply not only for the respondent giving you the information but also for their household members or other individuals who are included in the data.

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

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.[29]

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

Secure data storage and transfer are ultimately your personal responsibility.[30] 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

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

underlying data is obtained through a leak the information itself remains usable. Datasets that include confidential information *must* therefore be **encrypted**[31] during data collection, storage, and transfer. 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.

[31] **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

Chapter 3: Establishing a measurement framework

In this chapter we will show how you can save a lot of time and increase the quality of your research by planning your project's data requirements based on your research design. There are many published resources about the theories behind different research designs. This chapter will instead focus on how the design impacts a project's data requirements. We assume you have a working familiarity with the research designs mentioned here. If needed, you can reference Appendix XYZ, where you will find more details and specific references for each design.

Planning data requirements is more than just listing the data your project will need. It requires understanding how to structure the project's data to best answer the research questions, and creating the tools to share this understanding across your team. The first step of the process is to determine the data needs of the project, based on the research design and measurement framework. The data map diagrams each data source the project will use, the unit of response, frequency of measurement, and the level(s) at which it can be linked to other datasets within the framework. These master datasets will serve three key functions. First, they will list all of the units who are eligible for the study, and enable you to map data collected or received from the field to the research design. Second, in designs where your team has direct control over interventions or other field work, they will allow you to complete sampling and treatment assignment tasks before those tasks are implemented in the field. Finally, they will function as a single unambiguous location where all information related to the implementation and validity of your research is stored, as well as all information needed to correctly identify any observation in any of your project's datasets.

Almost all research designs rely on a random component for the results of the research to be a valid interpretation of the real world. This includes both how a sample is representative to the population studied, and how the counterfactual observations in experimental design are statistically indistinguishable from the treatment observations. The second half of this chapter covers random sampling and assignment and the necessary practices to ensure that these and other random processes are reproducible. It concludes with a discussion of power calculations and randomization inference, and how both are important tools to make optimal choices when planning data work.

Translating research design to master data

In most projects, different data sources are needed to answer the research question. These could be multiple survey rounds, data acquired from different partners (e.g. administrative data, web scraping, implementation monitoring, etc) or complex combinations of these. For example, you may have different **units of observation**[1], and their level may vary from round to round.

[1] The **unit of observation** is the unit at or for which data is collected. See https://dimewiki.worldbank.org/Unit_of_Observation

However your study is structured, you need to know how to link data from all sources and analyze the relationships between the units that appear in them to answer all your research questions. You might think that you are able to keep all of the relevant details in your head, but your whole research team is unlikely to have the same understanding, at all times, of all the datasets required. The only way to make sure that the full team shares the same understanding is to create *master datasets* and a *data map*.

Creating master datasets and a data map

A **master dataset**[2] details all project-wide time-invariant information about all observations encountered, as well as their relationship to the research design, typically summarized by sampling and treatment status. Having a plan for how to get the raw data into analysis shape before you acquire it, and making sure that the full research team knows where to find this information, will save you a ton of time during the course of the project and increase the quality of your research.

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

You should create a master dataset for each unit of observation relevant to the research. This includes all units used for significant research activity, like data collection or data analysis. Therefore, any unit that will be used in sampling or treatment assignment, must have a master dataset, and that master dataset – not field data – should be used when sampling or assigning treatment. Master data sets are often created from field data, but master data sets should be treated differently as there can be several field datasets, but only one authoritative master dataset. Master data sets should be created as soon as you start to get information about units. For example, when receiving a type of administrative data set for the first time, or after doing a respondent listing before a survey.

You also need to record how all datasets for each unit of observation will link or merge with each other as needed. This linking scheme is called a **data map**[3]. A data map is more than just a list of datasets. Its purpose is to specify the characteristics and linkages of those datasets. To link properly, the master datasets must include fully unique ID variables.[4] The master datasets should indicate whether datasets should be merge one-to-one, for example, merging baseline data and endline data that use the same unit of observation, or whether two datasets should be merged many-to-one, for example, school administrative data merged with student data. Your data map must indicate which ID variables can be used and how to merge datasets. It is common that administrative data use IDs that are different than the project IDs, and the linkage between those should be

[3] https://dimewiki.worldbank.org/data_map (TO BE CREATED)

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

clear from your master dataset.

The data map should also include metadata about the handling of all information. These characteristics may be updated as the project progresses. For example, you will need to note the original source of each dataset, as well as the project folder where the raw original data and codebooks are stored and where the back-ups for the each raw dataset are stored.

Some of the characteristics in your master datasets and your data map should be filled in during the planning stage, but both of them should be active resources that are updated as your project evolves. Finally, your master data should not include any unlabeled missing values. If the information is missing for one unit, then the reason should always be indicated with a code. An example for such reason could be that a unit was not included in the treatment assignment as it was not sampled in the first place, or was not located in the data collection at a given round.

Defining study comparisons using master data

Your research design will determine what statistical comparisons you need to estimate in your analytical work. The research designs discussed here compare a group that received some kind of **treatment**[5] against a counterfactual control group.[6] The key assumption is that each person, facility, or village (or whatever the unit of intervention is) had two possible states: their outcome if they did receive the treatment and their outcome if they did not receive that treatment. The average impact of the treatment, or the ATE[7], is defined as the difference between these two states averaged over all units.

However, we can never observe the same unit in both the treated and untreated state simultaneously, so we cannot calculate these differences directly. Instead, the treatment group is compared to a control group that is statistically indistinguishable, which makes the average impact of the treatment mathematically equivalent to the difference in averages between the groups. Statistical similarity is often defined as **balance** between two or more groups. Since balance tests are commonly run for impact evaluations, DIME Analytics created a Stata command to standardize and automate the creation of nicely-formatted balance tables: `iebaltab`[8]. Each research design has a different method for identifying the statistically-similar control group. for how the statistically similar control group is identified.

The rest of this section covers how data requirements differ between different research designs. What does not differ, however, is that the authoritative source for which units are in the treatment group and which are in the control group should always be one or

[5] **Treatment:** The general word for the evaluated intervention or event. This includes being offered training or cash transfer from a program, experiencing a natural disaster etc.

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

[7] The **average treatment effect (ATE)** is the expected average change in outcome that untreated units would have experienced had they been treated.

[8] `https://dimewiki.worldbank.org/iebaltab`

several variables in your master dataset. You will often have to merge that data to other datasets, but that is an easy task if you created a data map.

In **experimental research designs**, such as **randomized control trials (RCTs)**,[9] the research team determines which members of the studied population will receive the treatment. This is typically done by a randomized process in which a subset of the eligible population is randomly assigned to receive the treatment (see later in this chapter for how to implement this). The intuition is that if everyone in the eligible population is assigned at random to either the treatment or control group, then the two groups will, on average, be statistically indistinguishable. The treatment will therefore not be correlated with anything but the impact of that treatment.[10] The randomized assignment should be done using the master data, and the result should be saved there before being merged to other datasets.

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

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

Quasi-experimental research designs,[11] by contrast, are based on events not controlled by the research team. Instead, they rely on "experiments of nature", in which natural variation in treatment can be argued to approximate randomization. You must have a way to measure this natural variation, and how the variation is categorized as outcomes of a naturally randomized assignment must be documented in your master dataset. Unlike carefully planned experimental designs, quasi-experimental designs typically require the luck of having access to data collected at the right times and places to exploit events that occurred in the past. Therefore, these methods often use either secondary data, including administrative data or other classes of routinely-collected information, and it is important that your data map documents how that data is merged to any other data.

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

Regression discontinuity (RD)[12] 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. Common examples are test score thresholds and income thresholds, where the individuals on one side of some threshold receive a treatment but those on the other side do not.[13] The intuition is that, on average, individuals immediately on one side of the threshold are statistically indistinguishable from the individuals on the other side, and the only difference is receiving the treatment. In your data you need an unambiguous way to define which observations were above or below the cutoff. The cutoff determinant, or running variable, is often a continuous variable that is used to divide the sample into two or more groups. Both the running variable and a categorical cutoff variable, should be saved in your

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

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

master dataset.

Instrumental variables (IV)[14] designs, unlike the previous approaches, assume that the treatment effect is not directly identifiable. Similar to RD designs, IV designs focus on a subset of the variation in treatment take-up. Where RD designs use a *sharp* or binary cutoff, IV designs are *fuzzy*, meaning that the input does not completely determine the treatment status, but instead influence the *probability of treatment*. You will need variables in your data that can be used to estimate the probability of treatment for each unit. These variables are called **instruments**. In IV designs, instead of the ordinary regression estimator, a special version called two-stage least squares (2SLS) is used to estimate the impact of the treatment. Stata has a built-in command called `ivregress`, and another popular implementation is the user-written command `ivreg2`.

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

Matching[15] methods use observable characteristics to construct sets of treatment and control units where the observations in each set are as similar as possible. These sets can either consist of exactly one treatment and one control observation (one-to-one), a set of observations where both groups have more than one observation represented (many-to-many), or where only one group has more than one observation included (one-to-many). By now you can probably guess that the result of the matching needs to be saved in the master dataset. This is best done by assigning an ID to each matching set, and create a variable in the master dataset with the ID for the set each unit belongs to. The matching can even be done before the randomized assignment, so that treatment can be randomized within each matching set. This is a type of experimental design. Furthermore, if no control observations were identified before the treatment, then matching can be used to ex-post identify a control group. Many matching algorithms can only match on a single variable, so you first have to turn many variables into a single variable by using an index or a propensity score.[16] DIME Analytics developed a command to match observations based on this single continuous variable: `iematch`[17], part of the `ietoolkit` package.

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

[16] https://dimewiki.worldbank.org/Propensity_Score_Matching

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

Structuring complex data

Your data map and master dataset requirements also depends on whether you are using data from one time period or several. A study that observes data in only one time period is called a **cross-sectional study**. This type of data is relatively easy to collect and handle because you do not need to track individuals across time. Instead, the challenge in a cross-sectional study is to show that the control group is indeed a valid counterfactual to the treatment group.

Observations over multiple time periods, referred to as **longitudinal data**, can consist of either **repeated cross-sections** or **panel data**. In repeated cross-sections, each successive round of data collection uses a new random sample of observations from the treatment and control groups, but in a panel data study the same observations are tracked and included each round. If you are using panel data, then your data map must document how the different rounds will be merged or appended, and your master dataset will be the authoritative source of which ID that will be used.

You must keep track of the *attrition rate* in panel data, which is the share of observations not observed in follow-up data. It is common that the observations not possible to track can be correlated with the outcome you study. For example, poorer households may live in more informal dwellings, patients with worse health conditions might not survive to follow-up, and so on. If this is the case, then your results might only be an effect of your remaining sample being a subset of the original sample that were better or worse off from the beginning. You should have a variable in your master dataset that indicates attrition. A balance check using the attrition variable can provide insights as to whether the lost observations were systematically different compared to the rest of the sample.

Monitoring data

For any study with an ex-ante design, **monitoring data**[18] is very important for understanding whether field realities match the research design. Monitoring data is used to understand if the assumptions made during the research design corresponds to what is true in reality. The most typical example is to make sure that, in an experimental design, the treatment was implemented according to your treatment assignment.

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

Treatment implementation is often carried out by partners, and field realities may be more complex realities than foreseen during research design. Furthermore, the field staff of your partner organization, might not be aware that their actions are the implementation of your research. Therefore, you must acquire monitoring data that tells you how well the treatment assignment in the field corresponds to your intended treatment assignment, for nearly all research designs.

Another example of a research design where monitoring data is important are regression discontinuity (RD) designs where the discontinuity is a cutoff for eligibility of the treatment. For example, let's say your project studies the impact of a program for students that scored under 50% at a test. We might have the exact results of the tests for all students, and therefore know who should be offered

the program, however that is not the same as knowing who attended the program. A teacher might offer the program to someone that scored 51% at the test, and someone that scored 49% at the might decline to participate in the program. We should not pass judgment on a teacher that offers a program to a student they think can benefit from it, but if that was not inline with our research assumptions, then we need to understand how common that was. Otherwise the result of our research will not be helpful in evaluating the program.

Monitoring data is particularly prone to errors relating to merging with other data set. If you send a team to simply record the name of all people attending a training, then potentially different spellings of names – especially when names have to be transcribed from other alphabets to the Latin alphabet – will be the biggest source of error in your monitoring activity. The time to discuss and document how monitoring data will be merged with precision to the rest of your data, is when you are creating your data map. The solution is often very simple, it is just a matter of solving it before it is too late. An example of a solution is to provide the monitoring teams with lists of the people's names spelled the same way as in your master dataset. Include both the people you expect to attend the training, and people that you do not expect to attend, but do not tell the monitoring team which person you expect to attend the training.

Additionally, instruct the monitoring teams to record the name of any participant not in your lists. Add those names to your master datasets, as the most complete information possible will help you if you at any point in your project end up without an ID to merge on, and will have to compare names when merging data. Finally, while it is always better to monitor all activities, it might be to costly. In those cases you can sample a smaller number of critical activities and monitor them. This will not be detailed enough to be used as a control in your analysis, but it will still give an idea of the validity of your research design assumptions.

Implementing random sampling and treatment assignments

Random sampling and treatment assignment are two core elements of research design. In experimental methods, random sampling and treatment assignment directly determine the set of individuals who are going to be observed and what their status will be for the purpose of effect estimation. In quasi-experimental methods, random sampling determines what populations the study will be able to make meaningful inferences about, and random treatment assignment creates counterfactuals. Randomization[19] is used to ensure that a sample is representative and that any treatment and control groups

[19] **Randomization** is often used interchangeably to mean random treatment assignment. In this book however, *randomization* will only be used to describe the process of generating a sequence of unrelated numbers, i.e. a random process. *Randomization* will never be used to mean the process of assigning units in treatment and control groups, that will always be called *random treatment assignment*, or a derivative thereof.

are statistically indistinguishable after treatment assignment.

Randomization in statistical software is non-trivial and its mechanics are unintuitive for the human brain. The principles of randomization we will outline apply not just to random sampling and random assignment, but to all statistical computing processes that have random components such as simulations and bootstrapping. Furthermore, all random processes introduce statistical noise or uncertainty into estimates of effect sizes. Choosing one random sample from all the possibilities produces some probability of choosing a group of units that are not, in fact, representative. Similarly, choosing one random assignment produces some probability of creating groups that are not good counterfactuals. *Power calculation* and *randomization inference* are the main methods by which these probabilities of error are assessed. These analyses are particularly important in the initial phases of development research – typically conducted before any actual field work occurs – and have implications for feasibility, planning, and budgeting.

Randomizing sampling and treatment assignment

Sampling is the process of randomly selecting observations from a list of individuals to create a representative or statistically similar sub-sample. This process can be used, for example, to select a subset from all eligible units to be included in data collection when the cost of collecting data on everyone is prohibitive.[20] But it can also be used to select a sub-sample of your observations to test a computationally heavy process before running it on the full data. **Randomized treatment assignment** is the process of assigning observations to different treatment arms. This process is central to experimental research design. Most of the code processes used for randomized assignment are the same as those used for sampling, since it is also a process of randomly splitting a list of observations into groups. Where sampling determines whether a particular individual will be observed at all in the course of data collection, randomized assignment determines if each individual will be observed as a treatment observation or used as a counterfactual. The list of units to sample or assign from may be called a **sampling universe**, a **listing frame**, or something similar. This list should always be your **master dataset** when possible. The rare exceptions when master datasets cannot be used is when sampling must be done in real time – for example, randomly sampling patients as they arrive at a health facility. In those cases, it is important that you collect enough data during the real time sampling, such that you can create a master dataset over these individuals afterwards.

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

The simplest form of sampling is **uniform-probability random sampling**. This means that every eligible observation in the master dataset has an equal probability of being selected. 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. 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. This code uses a Stata built-in dataset and is fully reproducible (more on reproducible randomization in next section), so anyone that runs this code in any version of Stata later than 13.1 (the version set in this code) will get the exact same, but still random, results.

simple-uniform-probability-sampling.do

```
* Set up reproducible randmomization - 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 observations.
* 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 and 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 means that we
* have randomly sampled 20% of the observations.
    gen sample = (_n <= _N * 0.20)

* Restore the original sort order
    isid patient, sort

* Check your result
    tab sample
```

Sampling typically has only two possible outcomes: observed and unobserved. Similarly, a simple randomized assignment has two outcomes: treatment and control, and the logic in the code would be identical to the sampling code example. However, randomized assignment often involves multiple treatment arms which each represent different varieties of treatments to be delivered; in some cases, multiple treatment arms are intended to overlap in the same sample. Complexity can therefore grow very quickly in randomized assign-

ment 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 randomized assignment with multiple treatment arms.

simple-multi-arm-randomization.do

```
* Set up reproducible randmomization - 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 the two treatment arms
    generate treatment = 0                       // Set all observations to 0 (control)
    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
```

Programming reproducible random processes

For statistical programming to be reproducible, you must be able to re-obtain its exact outputs in the future.[21] We will focus on what you need to do to produce truly random results for your project, to ensure you can get those results again. This takes a combination of strict rules, solid understanding, and careful programming. This section introduces strict rules: these are non-negotiable (but thankfully simple). Stata, like most statistical software, uses a **pseudo-random number generator** which, in ordinary research use, produces sequences of number that are as good as random.[22] However, for *reproducible* randomization, we need two additional properties: we need to be able to fix the sequence of numbers generated and we need to ensure that the first number is independently randomized. In Stata, this is accomplished through three concepts: **versioning**, **sorting**, and **seeding**. We again use Stata in our examples, but the same principles translate to all other programming languages.

[21] 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

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

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 may be impossible to recover the original result. In Stata, the `version` command ensures that the list of random numbers is fixed.[23] The `ieboilstart` command in `ietoolkit` provides functionality to support this requirement.[24] We recommend you use `ieboilstart` at the beginning of your master do-file.[25] However, testing your do-files without running them via the master do-file may produce different results, since Stata's `version` setting expires after each time you run your do-files.

Sorting means that the actual data that the random process is run on is fixed. Because random numbers are assigned to each observation row-by-row starting from the top row, changing their order will change the result of the process. In Stata, the only way to guarantee a unique sorting order is to use `isid [id_variable], sort`. (The `sort, stable` command is insufficient.) 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 to increase during ongoing data collection), your randomization will not be reproducible 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.

Seeding means manually setting the start point in the list of random numbers. A seed is just a single number that specifies one of the possible start points. It should be at least six digits long and you should use exactly one unique, different, and randomly created seed per randomization process.[26] In Stata, `set seed [seed]` will set the generator to the start point identified by the seed. 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, change the sorting order, or alter the place of the random generator without you realizing it, so carefully confirm exactly how your code runs before finalizing it. To confirm that a randomization has worked correctly 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

[23] 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 the improvements do not matter in practice.

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

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

[26] 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 number on https://www.random.org.) There are many more seeds possible but this is a large enough set for most purposes.

any doubt that your results are reproducible. Once the result of a randomization is used in the field, there is no way to correct any mistakes.

reproducible-randomization.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 to check1 and check3
    graph matrix check1 check2 check3 , half
```

Some types of experimental designs require that randomized assignment 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, and as such are never reproducible. However, you can often run your randomization in advance even when you do not have list of eligible units in advance. Let's say you want to, at various health facilities, randomly select a sub-sample of patients as they arrive. You can then have a pre-generated list with a random order of "in sample" and "not in sample". Your field staff would then go through this list in order and cross off one randomized result as it is used for a patient.

This is especially beneficial if you are implementing a more complex randomization, for example, sample 10% of the patients, show a video for 50% of the sample, and ask a longer version of the questionnaire to 20% of both the group of patients that watch the video and those that did not. The real time randomization is much more likely to be implemented correctly, if your field staff simply can follow a list with the randomized categories where you are in control fo the pre-determined proportions and the random order. This way, you can also control with precision, how these categories are evenly distributed across all health facilities.

Finally, if this real-time randomization implementation is done

using survey software, then the pre-generated list of randomized categories can be preloaded into the questionnaire. Then the field team can follow a list of respondent IDs that are randomized into the appropriate categories, and the survey software can show a video and control which version of the questionnaire is asked. This way, you reduce the risk of errors in field randomization.

Clustering or stratifying a random sample or assignment

For a variety of reasons, random sampling and random treatment assignment are rarely as straightforward as a uniform-probability draw. The most common variants are **clustering** and **stratification**.[27] **Clustering** occurs when your unit of analysis is different than the unit of sampling or treatment assignment.[28] For example, a policy may be implemented at the village level, or you may only be able to send enumerators to a limited number of villages, or the outcomes of interest for the study are measured at the household level.[29] The groups in which units are assigned to treatment are called clusters. **Stratification** breaks the full set of observations into subgroups before performing randomized assignment within each subgroup.[30] The subgroups are called **strata**. This ensures that members of every subgroup are included in all groups of the randomized assignment process, or that members of all groups are observed in the sample. Without stratification, it is possible that the randomization would put all the members of a subgroup into just one of the treatment arms, or fail to select any of them into the sample. For both clustering and stratification, implementation is nearly identical in both sampling and randomized assignment.

Clustering is procedurally straightforward in Stata, although it typically needs to be performed manually. To cluster a sampling or randomized assignment, randomize on the master dataset for the unit of observation of the cluster, and then merge the results to the master dataset for the unit of analysis. This is a many-to-one merge and your data map should document how those datasets can be merged correctly. If you do not have a master dataset yet for the unit of observation of the cluster, then you first have to create that and update your data map accordingly. When sampling or randomized assignment is conducted using clusters, the clustering variable should be clearly identified in the master dataset for the unit of analysis 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.[31] This accounts for the design covariance within the cluster – the information that if one individual was observed or treated there, the

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

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

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

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

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

other members of the clustering group were as well.

By contrast, implementing stratified designs in statistical software is prone to error. Even for a typical multi-armed design, the basic method of randomly ordering the observations will often create very skewed assignments.[32] The user-written `randtreat` command properly implements stratification.[33] 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. Strata variables must also be included in your master datasets.

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

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

Doing power calculations for research design

Random sampling and treatment assignment are noisy processes: 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 randomized assignments 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 randomized assignments) 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 a true population parameter and the observed mean due to chance selection of units.[34] In randomized assignment, we call this the **randomization noise**, and define it as the difference between a 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 random sample or treatment assignment will fall in one of these categories, and we need to assess the likelihood and magnitude of this occurrence.[35] *Power calculation* and *randomization inference* are the two key tools to doing so.

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

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

Power calculations report the likelihood that your experimental design will be able to detect the treatment effects you are interested in given these sources of noise.[36] This measure of **power** can be described in various different ways, each of which has different practical uses.[37] The purpose of power calculations is to identify

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

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

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.

The **minimum detectable effect (MDE)**[38] is the smallest true effect that a given research design can reliably 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 small 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 never 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 implement a useful study.

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

Randomization inference[39] is used to analyze the likelihood that the randomized assignment 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 randomized assignment that did not in fact happen in reality. Randomization inference is particularly important in quasi-experimental designs and in small samples, where the number of possible *randomizations* is itself small. Randomization inference can therefore be used proactively during experimental design, to examine the potential spurious treatment effects your exact design is able to produce. If there is significant heaping at particular result levels, or if results seem to depend dramatically on the outcomes for 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.

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

Chapter 4: Acquiring development data

High-quality data is essential to most modern development research. Many research questions require novel data, because no source of publicly available data addresses the inputs or outcomes of interest for the relevant population. Data acquisition can take many forms, including: primary data generated through surveys; private sector partnerships granting access to new data sources, such as administrative and sensor data; digitization of paper records, including administrative data; web scraping; 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. 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. For data work to be reproducible, the data acquisition process must be clearly and carefully documented.

This chapter covers reproducible data acquisition, special considerations for generating high-quality survey data, and protocols for safely and securely handling confidential data. The first section discusses reproducible data acquisition: how to establish and document your right to use the data. This applies to all development datasets that are not publicly available, whether collected for the first time through surveys or sensors or acquired through a unique partnership. The second section goes into detail on data acquisition through surveys, as this process is typically more involved than acquisition of secondary data, and has more in-built opportunities for quality control. It provides detailed guidance on the electronic survey workflow, from questionnaire design to programming and monitoring data quality. We conclude with a discussion of safe data handling, providing guidance on how to receive, transfer, store, and share confidential 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

Clearly establishing and documenting data access is critical for reproducible research. This section provides guidelines for receiving data from development partners, establishing data ownership, and documenting the research team's right to use the data.

It is the researchers' responsibility to respect the rights of people who own the data and the people who are described by it; but also to make sure that information is as available and accessible as possible. 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.

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 an existing server, or physical access to extract certain information, or a one-time transfer of a block of data. In all cases, you must take action to ensure that data is transferred through secure channels so that confidential data is not compromised. See the section *Handling data securely* later in this chapter for how to do that. Keep in mind that compliance with ethical research standards may in some cases require a stricter level of security than initially proposed by the partner agency.

At this stage, it is very important to assess documentation and cataloging of the data and associated metadata. It is not always clear what pieces of information jointly constitute a "dataset", and many of the datasets you receive will not be organized for research. You should always retain the original data exactly 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 each file received. All too often data will be provided as vaguely-named spreadsheets, or digital files with non-specific titles, and documentation will be critical for future access and reproducibility. 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. DIME uses 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, the research team will need to decide the ap-

propriate level of aggregation for the data obtained. This dataset is the object you will think about cataloging, releasing, and licensing as you move towards publication. You may need to assess disclosure risk and seek clearance from the provider prior to publication, particularly if you are integrating various data sources.

Data ownership

Except in the case of primary surveys funded by the research team, the data is typically not owned by the research team. Rather, they must enter into data licensing agreements to access the data and publish derivative work. These agreements should make clear from the outset whether the research team can make the original data, any portion thereof, or derivatives[2] of the data, public.

Data ownership[3] can be challenging to establish, as many jurisdictions have differing laws regarding data and information, and the research team may be subject to multiple conflicting regulations. In some countries, 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 are varying norms. The best approach is always to consult with a local partner, and enter into specific legal agreements establishing ownership, access, and publication rights. This is particularly critical where confidential data is involved – that is, when people are disclosing information to you that you could not obtain simply by observation or through public records.

For original data generated by the research team, such as survey data, it is important to clarify up front who owns the data, and who will have access to the data. These details need to be shared with respondents when they are offered the opportunity to consent to participate in the study. If the research team is not collecting the data directly – for example, if a government, private company, or research partner is doing the data collection – make sure that you have an explicit agreement about who owns the resulting data. The contract for data collection should include specific terms as to the rights and responsibilities of each stakeholder. It must clearly stipulate which party owns the data produced, and that the research team maintains full intellectual property rights. The contract should also explicitly stipulate that the contracted firm is responsible for protecting respondent privacy, that the data collection will not be delegated to any third parties, and that the data will not be 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

[2] **Derivatives** of a data can be indicators, aggregates, visualizations etc. based on the data.

[3] need to add DIME Wiki page

standards for social science research, and adhere to the specific terms of agreement with the relevant Institutional Review Board or applicable local authority. Finally, it should include policies on reuse, storage, and retention or destruction of data.

Research teams that generate their own data must also consider data ownership downstream, the terms they will use to release that data to other researchers or to the general public. Will the team publicly release the data in full (removing personal identifiers)? Would the team be okay with it being stored on servers anywhere in the world, even ones that are owned by corporations or governments in other countries? Would the team prefer to decide permission on a case-by-case basis, dependent on specific proposed uses? Would the team expect that users of your data cite you or give you credit, or require users in turn to release their derivative datasets or publications under similar licenses? Whatever your answers are to these questions, make sure the agreement or contract you use to acquire the data specifically details those requirements.

Data licensing agreements

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

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[4]. 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.

[4] need to add a DIME wiki page

The World Bank has a template Data License Agreement which DIME follows.[5] The template specifies the specific objectives of the data sharing, and whether the data can be used only for the estab-

[5] https://worldbankgroup.sharepoint.com/teams/ddh/SiteAssets/SitePages/ddh/DataLicenseAgreementTemplate_v4.pdf?cid=68a54269-bbff-4b47-846d-cab248ad7de1

lished purpose or for other objectives consistent with the mandates of the signatory organizations. It establishes the type of future data access, using the following categories:

- open data: data freely available to the public
- licensed use: data available to the public by request; requesting users complete a licensing agreement with the Bank
- official use only: access limited to World Bank staff
- restricted access: access restricted to pre-authorized staff, or a specific business unit

The data provider may impose similar restrictions to sharing derivative works and any or all of the metadata. The template also specifies the required citation for the data.

Collecting data using electronic surveys

In this section, we detail specific considerations for acquiring high-quality data through surveys. 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 specific workflow considerations for digitally-generated data. 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. 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.[6] The focus of this section is the design of electronic field surveys, often referred to as Computer Assisted Personal Interviews (CAPI).[7] Although most surveys are now collected electronically, by tablet, mobile phone or web browser, **questionnaire design**[8] (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. 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.[9] It is essential to start with a clear understanding of the **theory of change**[10] and **research 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 the specific research design. The ideal starting point for this is a **pre-analysis plan**.[11]

Use the list of key outcomes to create an outline of questionnaire *modules*. Do not number the modules; instead use a short prefix as numbers quickly get outdated when modules are 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.

[6] 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

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

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

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

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

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

Each module should then be expanded into specific indicators to observe in the field.[12] At this point, it is useful to do a **content-focused pilot**[13] 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, and data analysis through the phases of a randomized trial.[14]

Once the content of the survey is drawn up, the team should conduct a small **survey pilot**[15] using the paper forms to finalize questionnaire design and detect any content issues. A content-focused pilot[16] 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.[17] 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.

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

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

[14] 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

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

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

[17] 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.[18] 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.[19] 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-

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

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

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**[20]. Second, if collecting data on medication use or supplies, you could collect: the brand name of the product; the generic name of the product; the coded compound of the product; or the broad category to which each product belongs (antibiotic, etc.). All four may be useful for different reasons, but the latter two are likely to be the most useful for data analysis. The coded compound requires providing a translation dictionary to field staff, but enables automated rapid recoding for analysis with no loss of information. The generic class requires agreement on the broad categories of interest, but allows for much more comprehensible top-line statistics and data quality checks. Rigorous field testing is required to ensure that answer categories are comprehensive; however, it is best practice to include an *other, specify* option. Keep track of those 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.

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

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.[21] 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`,

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

and so on.

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,[22] 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.

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

A second survey pilot should be done after the questionnaire is programmed. The objective of this **data-focused pilot**[23] is to validate the programming and export a sample dataset. Significant desk-testing of the instrument is required to debug the programming as

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

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.

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 data 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.

Handling data securely

All confidential data must be handled in such a way that only people specifically approved by an Institutional Review Board (IRB)[24] are able 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)[25] or that the data owner has specified restricted access.

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

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

Encryption

Data encryption is central to secure data handling. Data encryption 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.[26] 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 streamlined workflow, so that it is easy as possible to make sure the weakest link is still sufficiently secure.

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

Data that has been encrypted is made readable again using decryption, and decryption requires a password or a key. Encryption make data files completely unusable to anyone without the specific decryption key, which is a higher level of security than password-protection. It is never secure to share passwords or keys by email, WhatsApp or other insecure modes of communication; instead, use a secure password manager.[27] 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. Make sure you store your keys with descriptive names to match the encryption to which they correspond.

There are two main types of encryption algorithms and they are called **symmetric encryption**[28] and **asymmetric encryption**[29]. The only difference you need to know is that the same key is used to both encrypt and decrypt the data in symmetric encryption, while in asymmetric encryption, one key in a key pair is used to encrypt the data, and the other key in the same pair is used to decrypt it. We will show how this difference can be utilized for different use cases common in research.

Additionally, there are two contexts for encryption you should be aware of; **encryption-in-transit**[30] and **encryption-at-rest**[31]. Encryption-in-transit protects your data when in transit over the internet. It is easy to implement encryption-in-transit such that no input from the user is required, since the server and browser can keep track of the key for the short duration of the connection. All well established services have this implemented and you rarely have to worry about it. However, if you were to use a less well-known service, it is important that you verify that it use encryption-in-transit, as without it, your data and credentials are very vulnerable. Encryption-at-rest protects your data when it is stored somewhere, for example on a server or on your computer. In contrast to encryption-in-transit, you as a user, need to keep track of the key. Only people listed on your IRB is allowed to have access to this file. If you do not, the raw data will be accessible by

Many data-sharing and data-storage software providers will promote their services by advertising on-the-fly encryption and decryption. However, this is often only encryption-in-transit and not encryption-at-rest. Sometimes on-the-fly encryption involves encryption-at-rest, but then you have to verify how the encryption key is handled as individuals who are not approved by your IRB, such as tech support personnel, server administrators, and other third-party staff are not allowed to have access to the key. It is possible, in some enterprise versions of data sharing software, to set up appropriately secure on-the-fly encryption. However, that setup is

[27] https://lastpass.com or https://bitwarden.com among many other. See DIME Analytics' guide to password managers here: https://github.com/worldbank/dime-standards/blob/master/dime-research-standards/pillar-4-data-security/data-security-resources/password-manager-guidelines.md

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

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

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

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

advanced, and you should never trust it unless 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, where you set up your own encryption where you are in full control of who has access to the key.

Storing data securely

The first thing you need to do before planning how to securely receive data, is to plan how to securely store data after you have received it. Typically, you want to store your data so that you can decrypt and access it, interact with it and then decrypt it again. That is a perfect use case for **symmetric encryption** where the same key is used to both encrypt and decrypt your files. Think of this as a physical safe where you have one key used to both add content and access content.

The equivalent to a safe in secure data storage is an encrypted folder, which you can set up using, for example, VeraCrypt.[32] Files in an encrypted folder can be interacted with and modified like any unencrypted file, as long as you have the key. This is an implementation of encryption-at-rest. There is absolutely no way to restore your data if you were to lose your key, so we cannot stress enough the importance of using a password manager, or equally secure solution, to store your keys.

[32] https://www.veracrypt.fr and https://github.com/worldbank/dime-standards/blob/master/dime-research-standards/pillar-4-data-security/data-security-resources/veracrypt-guidelines.md

It is becoming more and more common that development research is done on data set that is too big to store on a regular computer, and instead the data is stored and processed in a cloud environment. There are many available cloud storage solutions and you need to understand how the data is encrypted and how the keys are handled. This is likely another case where a regular research team will have to ask a cybersecurity expert. After someone have helped you to set up a secure cloud storage, if you were to download a sample of the data – for example to develop your code on – then you need to remember to encrypt the data when stored on your computer.

Collecting data securely in the field

All common data collection software will automatically encrypt all data in transit (i.e., upload from field or download from server). However, it is also necessary to ensure that confidential data are protected when stored on a server owned by the data collection software provider. You should not assume that your data is encrypted-at-rest by default. In most data collection platforms, encryption-at-rest needs to be explicitly enabled and operated by the user.

When collecting data you want the tablets or the browsers used for

data collection to be able to encrypt all information before submitting it to the server. At the same time you do not want those devices to have any way of decrypting already submitted data. This is a perfect use case for **asymmetric encryption** where there are two keys in a public/private key pair. The public key can safely be sent to all tablets or browsers to be used for encrypting the data. Only the private key in the same key pair can then be used to decrypt that data. The private key should be kept secret and should not be shared with anyone not listed on the IRB. Again, you must store the key pair in a secure location, such as in a secure note in a password manager, as there is no way to access your data if the private key is lost.

Data encrypted this way can securely pass through the server owned by the data collection software provider, and you should never decrypt it until it is downloaded on your computer. If your data collection service allows you to browse data in the browser, then the encryption is only implemented correctly if you are asked for the key each time.

Finally, the data security standards that apply when receiving confidential data from the field also apply when transferring confidential data to the field. In some survey software, you can use the same encryption that allows you to receive data securely from the field, to also send confidential data, such as an identifying list of respondents, to the field.

Receiving or sharing data securely

If you receive confidential data from a partner organization or share confidential data with your research team over the internet, it must always be encrypted. The best practice is to reduce the amount of confidential data needed to be shared or transfer. If you do not need the confidential information you will simplify your workflow by collecting or receiving data without any confidential information in the first place. This should always be your first option when possible.

Usually we need to deal with confidential data at some point in the project, and then the second best practice is create two versions of your data. One version of your data will still have the confidential information, and it will need to be encrypted at all times. This version should only be used when the confidential information is actually needed. In the second version of your data all confidential data should be removed. This version does not have to be encrypted and should be what you use on a day-to-day basis.

The most common reason a data is confidential is that it includes personal identifiers. Removing identifying information is called de-identification, and we will discuss how to do that in the next

chapter. De-identifying the data at the earliest possible opportunity will simplify the workflow without compromising data security, as de-identified data no longer needs to be encrypted. Another method to protect confidential data is to aggregate the data, meaning that sums and averages of group of observations are calculated and only those numbers are reported. Any of these techniques can be used either before the data is shared with the research team, or when creating the second non-confidential version of the data set to be used in the analysis. If confidential information is directly required for the analysis, it will be necessary to keep at least a subset of the data encrypted through the data analysis process.

There are multiple options for sharing encrypted data. If you need to collaborate on a confidential data set then you can create an encrypted folder, for example using VeraCrypt (as described above). An encrypted folder can safely be shared using insecure modes of communication such as email or third-party syncing services. Anyone who has access to both the encrypted folder and the encryption key can read and collaborate on the encrypted file.

If you only need a one-time transfer of confidential information, such as to receive a data set from a partner organization, then setting up a permanent encrypted folder may be unnecessary. Instead, for one-time transfers we recommend sending data over password-protected end-to-end encrypted file sharing services.[33]

[33] One such service at the time of writing is https://send.firefox.com/

Backing up original data

While encryption is protecting your data from unauthorized access, you also need to protect your data from being accidentally overwritten or deleted. This is done through a back-up protocol. This is an example of such a protocol:

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

2. Download your original data from your data source to that encrypted folder. If your data source is a survey and the data was encrypted during data collection, then you will need *both* the private key used during data collection to be able to download the data, *and* the key used when you created the encrypted 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 second encrypted folder on an external drive that you can keep in a secure location. Copy the data you just downloaded to this second encrypted folder. This is the "master" backup copy

of the raw data. You should never work with this data on a day-to-day basis. You should not use the same encrypted folder or the same key as above, because if you use the same key and lose the key, then you will have lost access to both encrypted folders. If you have an physical safe where you can securely store the external drive, then you do not need to encrypt the data and thereby do not risk losing access by losing an encryption key.

4. Finally, create a third encrypted 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 second location, for example, at another office of your organization. This is the "golden master" backup copy of the raw data. You should never store the "golden master" copy in a synced folder, as it would be deleted in the cloud storage if it is deleted on your computer. You should also never work with this data on a day-to-day basis.

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. 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.

Once the raw data is securely stored and backed up, it can be transformed into your final analysis dataset through the steps described in the next chapter.

Chapter 5: Cleaning and processing research data

Raw data files may be received in a variety of formats, and are often not easy to handle using statistical software. The tasks described in this chapter have many different names. Some say data cleaning, others data munging, or maybe data wrangling. But they all mean the same thing: transforming raw data into a more convenient format for your intended use. This is the most time-consuming step of a project's data work, particularly when primary data is involved. It is also essential for data quality. This chapter outlines the workflow our team developed to make the process efficient and transparent. One key point of this workflow is that no changes be made to the contents of data at this point. Creating new variables, imputing values and correcting outliers are research decisions. To make data processing less error-prone, such changes to the data will be made at a separate stage. Therefore, the clean dataset, which is the main output from the tasks discussed in this chapter, contains almost exactly same information as the raw data, but in a format that is easier to handle.

The following sections describe the different tasks involved in making newly acquired data ready for analysis. The first section will show how to make your data *tidy*. This means adjusting how the dataset is organized until the relationship between its rows and columns is well-defined. The second section describes quality assurance checks, which are necessary to verify the accuracy of your data. After that, identifying information is typically no longer needed, therefore the third section outlines steps to remove direct identifiers. Finally, the last section discusses how to explore each variable in your dataset and make sure that it is as well documented and as easy to use as possible.

Identifying the identifier(s)

The very first step in creating analysis-friendly datasets is understanding the data acquired, and translating this understanding into an intuitive format. This section discusses what steps may be needed to make sure that each row in a tabular dataset[1] represents one observation. This may seem trivial, but should not be taken for granted. Getting to such a format may be harder than expected, and the **unit of observation**[2] may be ambiguous in many raw datasets. In this section, we will discuss the ideal format to handle data, which we call a *tidy* format.

[1] **Tabular data:** data that is structured into rows and columns. Also called "rectangular data". Examples of non-rectangular data are a text, or files, such as images.

[2] **Unit of observation:** the unit described by the data. In datasets, it is ideally what each row represents. https://dimewiki.worldbank.org/wiki/Unit_of_Observation

Unique identifiers

The first step in tidying data is to understand its unit of observation (what makes a row), and find out which variable or set of variables is the **unique identifier**[3] for each observation in your dataset. Ensuring that observations are uniquely and fully identified is arguably the most important step in data cleaning. It may be the case that the variables expected to be unique identifiers are either incomplete or contain duplicates. It is also possible for a dataset to have no unique identifier, or that the identifier involves a long string that is difficult to work with, such as a full name. In such cases, cleaning begins by carefully creating a variable that uniquely identifies the data. Note that while digital survey tools create unique identifiers for each data submission, that is not the same as having a unique ID variable for each observation in the sample, as there can be multiple submissions for the same observation. The unique identifier will be used to link data on observations in one dataset with data on the same observations in other datasets, such as other rounds of data collection and different data sources. Therefore, you must keep a canonical dataset of all observations in your data. This **master dataset**[4] is discussed in detail in Chapter 3, and contains a record of all units ever encountered for each relevant level of observation, as well as their corresponding unique identifiers.

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

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

`ieduplicates` and `iecompdup`, two Stata commands included in the `iefieldkit` package,[5] create an automated workflow to identify, correct and document occurrences of duplicate entries. One advantage of using `ieduplicates`[6] to correct duplicated entries is that it creates *duplicates reports* that record which corrections made and why. Even if you are not using this command, it is important to keep a record of all cases of duplicated IDs encountered.

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

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

Tidying raw data

Though raw data can be acquired in all shapes and sizes, development data is most commonly received as one or multiple tables. Similarly, these tables can organize information in multiple ways, and not all of them result in easy-to-handle datasets. Fortunately, a vast literature of database management has identified the format that makes interacting with the data as easy as it can be. In database management this is called *normalization*, but when applied to statistics, we call data in such format **tidy**.[7] A dataset is tidy when each column represents one variable,[8] each row represents one observation, and all variables in a table have the same unit of observation. Every other format is *untidy*. This may seem trivial, but raw data, and raw survey data in particular, is rarely received in a tidy format. To contribute to

[7] **Tidy data:** Datasets where each column represents one variable, each row represents one observation, and all variables in a table have the same unit of observation.

[8] **Variable:** the collection of all data points that measure the same attribute.

the confusion, in Stata each column is called a "variable", and each row an "observation", even though they are not necessarily the same thing.

Every untidy dataset is untidy in its own way, but the most common case of untidy raw data encountered in development research are datasets consisting of multiple units of observation stored in the same table. If your rows include multiple nested observational units, having an identifying variable for each row doesn't mean having an identifier for each observation. Survey data containing nested units of observation is typically imported from survey platforms in **wide format**.[9] Take, for example, a household survey that includes household-level questions, as well as a household member roster. The resulting raw dataset usually consists of a single table where questions from the household member roster are saved in different columns, one for each member, with a corresponding member suffix, and household-level questions are represented by one column each. So you have one column called `ownsmotorcycle`, but a few columns of the format `gender_1`, `gender_2`, etc. The data is saved in this format because it's the most efficient way to transfer it: adding different levels of observation into the same table allows for data to be transferred in a single file. However, this leads to the widespread practice of interacting with data in wide format for a large part of a project, although this is not the most efficient way to do so.

[9] **Wide data:** a table where a single variable is divided into multiple columns, for example one for each individual in a household.

To understand how dealing with wide data can be complicated, just suppose you need to calculate the share of women in your sample using the dataset described above. The first thing you will need to do is change the format of your table so that each row represents one individual. Tidy datasets, on the other hand, are easier to handle because observations can be aggregated and linked consistently, without additional steps. They are also easier to clean, as each attribute only needs to be checked once, as corresponds directly to one question in the questionnaire.

The only example of untidy formats we mentioned so far is wide data, but as mentioned earlier, there are unlimited ways for data to be untidy. An alternative example is a table containing information on transactions and on the firms involved in each transaction. In this case, the firm-level information will be repeated for all transactions a given firm was involved in. Analyzing data in this format would give more weight to firms that conducted more transactions, which may not be consistent with your research design.

The basic process behind tidying a dataset is simple: first, you need to identify all the variables that were measured at the same level of observation; then, create separate tables for each level of observation; finally, reshape[10] the data and remove duplicated rows

[10] **Reshape:** transform a dataset in such a way that the unit of observation represented by a row changes.

until each table is uniquely and fully identified by the ID variable that corresponds to its unit of observation. In the earlier household survey example, household-level variables will be stored in one tidy dataset, and household-member variables are reshaped to member level and stored in another tidy dataset, which also contains the household ID for each individual. So the household ID is intentionally duplicated in the household members table. In the transaction data example, the result of the tidying process would be one transaction-level dataset, containing variables indicating the ID of all firms involved; and one firm-level dataset with a single entry for each firm.

Reshaping tables is the most intricate task in this process, so you should be very familiar with commands such as `reshape` in Stata and `pivot` in R. Another key aspect in this process is to make sure that ID variables are consistent across tables, so they can always be linked. There must be a clear way to connect each tidy table to a master dataset. Note that the tidying process gets more complex as the number of nested groups increases. That means the steps of identifying the unit of observation of each variable and reshaping the separated tables need to be repeated multiple times. However, the more nested groups a dataset includes, the more efficient it is to deal with tidy data as compared to untidy. Cleaning and constructing wide datasets, in particular, is a repetitive and error-prone process.

The next step of data cleaning, data quality monitoring, may involve comparing different units of observation. Aggregating subunits to compare to a higher unit is much easier with tidy data, which is why we suggest getting your data into a tidy format as early as possible in your workflow. If you are conducting primary data collection, you can start coding the data tidying before the data is acquired, since you will know in advance in which format it will be received. In the case of survey data, tidying datasets will guarantee a one-to-one correspondence between questions in the questionnaire and columns in the data. Preparing the data for analysis, the last task in this chapter, is much simpler when that is the case.

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. Although original data typically requires more extensive checks than secondary data, you should carefully examine and clean any data you are about to use. When reviewing raw data, you will inevitably encounter data entry mistakes, such as typos and inconsistent values. 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 acquisition preparation, the research team should develop a **data quality assurance plan**[11]. While data acquisition 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.

[11] 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 study 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. The faster issues are identified, the more likely they are to be solved. `ipacheck`[12] is a very useful command that automates some of these tasks, regardless of the source of the data.

[12] 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 received 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 ensure completeness of the 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.[13] `ieduplicates`[14] 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

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

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

for monitoring attrition, so that it is clear early on if a change in protocols or additional tracking will be needed. It is also important to check data collection completion rates and sample compliance by surveyors and survey teams, 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 incorporate 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. A retailed knowledge of the questionnaire and a careful examination of the analysis plan is the best preparation. Examples include verifying consistency across multiple response fields or data sources, 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.[15] 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.

[15] 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 provider. 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 platform. 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**[16] and other validation audits

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

help ensure that data collection is following established protocols, and that data is not falsified, 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.

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 validation 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.[17] 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.

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

De-identifying research data

Up to this point, you were probably handling confidential data. This is due to the fact that effective data quality monitoring frequently requires you to identify the individual observations in your dataset, and the people who acquired information about them. Having this data will allow you to quickly follow up on issues identified, and verify what caused them. Handling confidential data such as **personally-identifying information** requires a secure environment and often involves decrypting data before accessing it. De-identifiying the data simplifies this workflow and reduces the risk of harmful leaks. This section describes how to perform de-identification of data to be shared within a research team and with a wider audience.

Protecting privacy as a researcher

Most of the field research done in development involves human subjects.[18] As a researcher, you may have access to personal information

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

about your subjects: where they live, how rich they are, whether they have committed or been victims of crimes, their names, their national identity numbers, and other sensitive data. PII data carries strict expectations about data storage and handling, and it is the responsibility of the research team to satisfy these expectations.[19] Your funder or employer will most likely require you to hold a certification from a source such as Protecting Human Research Participants[20] or the CITI Program;[21] and your team will need to present a plan to handle data securely for IRB approval.

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

[20] https://phrptraining.com

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

In general, though, you shouldn't need to handle PII data very often once data collection is 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 their specific tasks. Data analysis that requires identifying information is rare and in most cases can be avoided by properly linking masked identifiers to research information such as treatment statuses and weights, then removing unmasked 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.[22] 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 identifying individuals and the utility of the data before publicly releasing a dataset.[23] The following section describes how to implement these processes.

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

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

De-identification in practice

To simplify workflows, the data should be de-identified as early as possible. Once you create a de-identified version of the dataset, you no longer need to interact directly with the encrypted raw data. Note that if the raw data contained different units of observation, at this

point your dataset should consist of set of tidy tables. In this case, each of them needs to be de-identified separately, but the workflow will be the same for all of them.

During the initial round of de-identification, datasets must be stripped of personally identifying information.[24] To do so, you will need to identify all variables that contain such information.[25] For data collection, where the research team designs the survey instrument, flagging all potentially identifying variables at 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` and IPA's `PII_detection`[26], scan variable names and labels for common string patterns associated with identifying information.[27] The World Bank's `sdcMicro` lists variables that uniquely identify observations, but it is less efficient for large datasets.[28] 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.[29]

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

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

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

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

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

[29] 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 is strictly required for the analysis itself and cannot be masked or encoded, it will be necessary to keep at least a subset of the data encrypted through the data analysis process.

After the this step, your dataset will consist of one or multiple tidy, de-identified tables. This is the underlying source for all cleaned and constructed data, the dataset that you will interact with directly during the remaining tasks described in this chapter. Because iden-

tifying 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. However, access to this data should still be restricted to the research team only, as it's possible that combining a group of variables, or taking into account information that is not the dataset, may lead to the identification of the individuals in the sample. It is common, and even desirable, for teams to make data publicly available once the tasks discussed in this chapter are concluded. This will allow other researchers to conduct additional analysis and to reproduce your finding. Before that can be done, however, you should further consider whether your data can be re-identified, in a process we call **final de-identification**, which will be discussed in more detail in chapter .

Preparing data for analysis

The last step in the data cleaning process involves making the dataset easy to use and understand, and carefully exploring each variable to document their distributions and identify patterns that may bias the analysis. The resulting dataset will contain only the variables collected in the field, and no modifications to data points will be made, except for corrections of mistaken entries. You may have more tables in your dataset now then originally received, and they may have a different *format*, but the information contained is still the same. Apart from the **cleaned dataset** (or datasets) itself, cleaning will also yield extensive documentation describing it.

During data cleaning, you will acquire in-depth understanding of the contents and structure of your data. This knowledge will be key to correctly construct final indicators and analyze them. So don't rush through this step. It is common for cleaning to be the most time-consuming task in a project. In this section, we will introduce some concepts and tools to make it more efficient and productive. The section is separated into three topics: exploring the data, making corrections, and recoding and annotating. They are separated here because they are different in nature. In practice, however, they are all done at the same time.

Getting to know your data

The first time you interact with your data is during quality checks. However, these checks are often programmed before you receive all the data. Furthermore, because quality checks are usually time-sensitive, there may not be time to explore the data at length. During data cleaning, on the other hand, you will need to inspect each

variable closely. Use tabulations, summary statistics, histograms and density plots to understand the structure of data, and look for patterns or irregularities. Think critically about what you see: are the values shown consistent with the information the variable represents? How do distributions look? Are there any outliers or missing values? Could distributional patterns be caused by data entry errors? Are related variables consistent with each other?

At this point, it is more important to document your findings than to try to address any irregularities found. There is a very limited set of changes that should be made to the raw data during cleaning. They are described in the next two sections, and are usually applied to each variable as you explore them. Most transformation resulting in data points that were not directly observed in the original data will be done during **construction**, a process discussed in the next chapter. For now, focus on creating a record of what you observe, extensively documenting the data being explored. You will use this documentation when discussing with your team how to address irregularities when you get to the construction stage. This material will also be valuable during exploratory data analysis.

Correcting data points

As mentioned earlier, corrections to issues identified during data quality monitoring are the only changes done to individual data points during the data cleaning stage. However, there is a lot of discussion about whether one should modify such data points at all. Some argue that follow-ups to the issues identified are costly and add limited value. Since it is not possible to check each and every possible data entry error, doing so can create a false sense of security from issues identified on a few main variables. Additionally, manually inspected data may suffer from considerable inspector variability. Therefore, there's an argument for using data quality monitoring tools as a tool to detect completely fraudulent observations and revisiting data collection protocols or decisions. On the other hand, there is also an argument to be made against keeping clear typing errors that have been found or for collecting data about mistakenly skipped survey modules.

Whether you decide to modify your data or not, a careful record of which data entry mistakes were identified. If no data points are modified, it may still be helpful to add flags to observations containing potentially problematic values, so you can verify how they affect results during analysis. If your team decides to follow up on these issues and try to determine what the correct value is, the follow-up process must also be thoroughly documented. 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. Finally, remember not to make changes directly to the raw data. Instead, any corrections must be done as part of data cleaning, and saved to a new dataset.

Recoding and annotating data

The cleaned dataset is the starting point of data analysis. It will be extensively manipulated to construct analysis indicators, so it is important for it to be easily processed by statistical software. To make the analysis process smoother, anyone opening it for the first time should have all the information needed to interact with it, even if they were not involved in the acquisition or cleaning process. This will save them time going back and forth between the dataset and its accompanying documentation.

Often times, datasets are not imported into statistical software in the most efficient format. The most common example is string (text) variables: categorical variables and open-ended responses are often read as strings. However, variables in this format cannot be used for quantitative analysis. Therefore, categorical variables must be transformed into other formats, such as `factors` in R and `labeled integers`[30] in Stata. Additionally, open-ended responses stored as strings usually have a high risk of being identifiers, so cleaning them requires extra attention. The choice names in categorical variables (called *value labels* in Stata and *levels* in R) should be accurate and concise, and link directly to the data collection instrument. Adding choice names to categorical variables 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.

[30] https://dimewiki.worldbank.org/wiki/Data_Cleaning#Value_Labels

In survey data, it is common for non-responses such as "Don't know" and "Declined to answer" to be represented by arbitrary survey codes. The presence of these values could bias your analysis, since they don't represent actual observations of an attribute. They need to be turned into *missing values*. However, the fact that a respondent didn't know how to answer a question is also useful information that would be lost by simply omitting all information. In Stata, this information can be elegantly conserved using extended missing values.[31]

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

We recommend that the cleaned dataset be kept as similar to the raw data as possible. This is particularly important regarding variable names: keeping them consistent with the raw data makes data processing and construction more transparent. Unfortunately, not all variable names are informative. In such cases, one important

piece of documentation, the variable dictionary, makes the data easier to handle. When a data collection instrument is available, it is often the best dictionary one could ask for. But even in this cases, going back and forth between files can be inefficient, so annotating variables in a dataset is extremely useful. In Stata, *variable labels*[32] must always be present in a cleaned dataset. They should include a short and clear description of the variable. A lengthier description, that may include, for example, the exact wording of a question, may be added through *variable notes*. In R, it is less common to use variable labels, and a separate dataset with a variable dictionary is often preferred, but `dataframe attributes` can be used for the same purpose.

[32] https://dimewiki.worldbank.org/wiki/Data_Cleaning#Variable_Labels

Finally, any information that is not relevant for analysis may be removed from the dataset. In primary data, it is common to collect information for quality monitoring purposes, such as notes, duration fields and surveyor IDs. Once you are past the quality monitoring phase, these variables may be removed from your dataset. In fact, to make the data easier to handle, you may choose to start from a minimal set of variables, and add new ones as you clean them. 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.

Although all these tasks are key to making the data easy to use, implementing them can be quite repetitive and create convoluted scripts. The `iecodebook` command suite, part of the `iefieldkit` Stata package, is designed to make some of the most tedious components of this process more efficient.[33] It also creates a self-documenting workflow, so your data cleaning documentation is created alongside that code, with no extra steps. As far as we know, there are no similar resources in R. However, the `tidyverse`[34] packages compose a consistent and useful grammar to perform the same tasks.

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

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

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,[35] and should be stored alongside your data dictionary and codebooks. You will probably need these files

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

during analysis, and they should be published along with the data, so other researchers may use them for their analysis as well.

Once you have a cleaned, de-identified dataset and the documentation to support it, you have created the first research output of your project: a dataset. Chapter 7 will get into the details of data release. For now, all you need to know is that your team should consider submitting this dataset for publication, or at least archiving and cataloguing, 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.

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 data originally acquired 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 the data analysis workflow pays substantial dividends throughout the process.

In this chapter, we suggest an analytical workflow starting from clean raw data. We first cover variable construction: this is when the clean data is transformed into the actual indicators that will be used for analysis. We emphasize the importance of transparently documenting research choices made during construction. We then 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 to make sure it is easy to follow and understand. Finally, we discuss options to automate common outputs so that your work is fully reproducible.

Constructing analysis datasets

For this chapter, we assume you are starting from one or multiple tidy,[1,2] well-documented datasets that have gone through thorough quality checks, and that incorporate any corrections needed. No matter where you received your data from, and how well-structured it was, in the previous chapter you can find all the steps needed to make sure it is now easy to use. The next step is to **construct**[3] the dataset, or datasets, you will use for analysis; that is, to transform the cleaned data into analysis-ready data. This is accomplished by integrating different datasets and creating derived variables (dummies, indices, and interactions, to name a few). The derived indicator you need to construct should be planned during research design, so the pre-analysis plan can serve as a guide. During this process, the data will typically be reshaped, merged, and aggregated to change the level of the data points from the unit of *observation* in the original data to the unit of *analysis* in the constructed data.[4]

A constructed dataset is built to answer an analysis question.

[1] Wickham, H. and Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 1st edition

[2] Availableathttps://vita.had.co.nz/papers/tidy-data.pdf

[3] **Data construction**: The process of transforming cleaned data into analysis data by creating the derived indicators that will be analyzed.

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

Since different pieces of analysis may require different sub-samples, or even different units of observation, you may have one or many constructed datasets, depending on your analysis plan. You often cannot create a single "canonical" analysis dataset. It is common to have many purpose-built analysis datasets. For a concrete example of what this means, 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 a single analysis dataset that constantly needs to be transformed.

Fitting construction into the data workflow

Construction is done after from data cleaning for two reasons. First, to clearly differentiate correction of data entry errors (necessary for all interactions with the data) from creation of analysis indicators (necessary only for specific analyses). Second, to ensure that variable definitions are consistent across. Construction can create many outputs combining many inputs, and, unlike in data cleaning, output datasets should be parsimonious rather than exhaustive. For example, take a project that has a baseline and an endline survey. Unless the two data collection instruments are exactly the same, which is preferable but often not the case, the data cleaning for each of these rounds will require different steps, and therefore will be done separately. However, the analysis indicators must be constructed in the exact same way, so they are comparable. To do this, you will require at least two separate cleaning scripts, and a unified construction script. Maintaining one construction script guarantees that you will not accidentally make changes to an indicator from one round while forgetting to update the other.

Construction of analysis data should be done right data cleaning and before data analysis, according to the analysis plan. In practice, however, as you analyze the data, different constructed variables often become necessary, as well as different subsets of the sample and other transformations of the data. So you will need to adjust the analysis data accordingly. But 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 and data transformation in a unified script, separate from the analysis code, helps avoid this and ensures consistency across different outputs.

Integrating different data sources

Combining or merging information from different data sources is often times necessary 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 include location-specific controls.[5] To understand how to perform such operations, you will need to consider the unit of observation for each dataset, and their respective identifying variables. Merges are common sources of missing values, and can also result in observations being dropped. To avoid unintentional changes to your data, pay close attention to merge results, and add checks to your code verifying that they match what you expect.

[5] Such operations are commonly called "merges" in Stata, and "joins" in R's `tidyverse` dialect. We will use the first term on the next few paragraphs.

If the datasets you need to merge have the same unit of observation, merging may be straightforward. The simplest case is when the datasets have the same unit of observation and use a consistent, uniquely and fully identifying ID variable. For example, in the case of administrative data for firms, you may merge data from different years using the firm taxpayer code (or, preferably, a deidentified version of it). In such cases, the main source of unexpected results are unmatched observations. Fortunately, it is straightforward to avoid mistakes in this setting: before writing code to combine the datasets, write pseudocode to understand which observations you expect to be matched or not, and why. When possible, determine exactly which and how many matched and unmatched observations should result from the merge. For example, first that were created in the second year should not be present in the first year's data, and firms that closed during the first year should not be observed in the second year's data. Then think carefully about whether you want to keep matched and unmatched observations, or only specific matching outcomes, for example, to create a balanced panel, and add that to your pseudocode as well. Now you can merge the datasets, and test that the outcome matches your expectations. Add comments to explain any exceptions, and make it so the code will return an error in case unexpected results show up in future runs. This approach can be extended to other types of merges between two related datasets.

Even when matching data sets with the same unit of observations, however, more complex cases may arise. Datasets at the same unit of observation may not use a consistent identifier. One common exam-

ple is for dataset identifiers to be come in the form of names, which are often spelled differently in different data sources. In these cases, you will need to do a **fuzzy match**, to link observations that have similar identifiers. You will need to analyze the merging patterns extensively, understand what units are present in one dataset but not the other, and investigate unmatched observations to resolve imperfect matching. There are some commands, such as `matchit`[6] in Stata and `agrep`[7] in R, that can provide useful utilities. However, more often than not, a large amount of manual examination is necessary in order to figure out what the matching pattern should be and how to accomplish it in practice through your code.

[6] https://ideas.repec.org/c/boc/bocode/s457992.html

[7] https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/agrep

Integrating datasets with different units of observation may involve changing the structure of the data. This often involves changing the unit of observation through aggregation or reshapes. For example, you may have data on individual students in a school, and wish to create a summary of the student-level data set to merge to the school data, including variables such as gender proportions or average family income. On the other hand, you may have asked farmers what quantity they harvested of different crops, and need to reshape these observations to the crop level to merge with data on crop prices. Both operations should always be done with great care. Two issues to pay extra attention are missing values and dropped observations. Make sure to read about how each command treats missing observations: are unmatched observations dropped? Are missing values turned treated as zero when aggregated? 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, prefer to drop observations explicitly, indicating why you are doing so and how the data set changed.

Note, however, that not all merges of data with different units of observation are as conceptually straightforward. More complex cases may arise if you are looking to overlay road location data with household data, using a spatial match; to combine school administrative data, such as attendance records and test scores, with household demographic characteristics from a survey; or link a dataset of infrastructure access points, such as water pumps or schools, and a dataset of household locations. In these cases, a key part of the research contribution is figuring out what a useful way to combine the datasets is. Since the conceptual constructs that link observations from the two data sources are so important and can take many possible forms, it is especially important for the data integration to not be treated mechanically, and to be extensively documented separately from other data construction tasks.

Creating analysis variables

Once you have assembled variables from different sources into a single working dataset with the right raw information and observations, it's time to create the derived indicators of interest for analysis. Before constructing new indicators, you must check and double-check units, scales and value assignment of each variable that will be used. This is when you will use the knowledge of the data and the documentation developed during cleaning the most. 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 `gender` 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. 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.

At this point, you will also need to decide how to handle any outliers or unusual values identified during data cleaning. How to treat outliers is a research question. There are multiple possible approaches, and the best choice for a particular case will depend on the objectives of the analysis. Whatever your team decides, make sure to explicitly note what the decision was and how it was made. 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. All these points also apply to imputation of missing values and other distributional patterns. As a general rule, never overwrite or delete original data during the construction process.

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 some analysis 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. Our team created `iefieldkit`'s `iecodebook append` subcommand to help you reconcile and append data from cleaned survey rounds. This is done by filling an Excel sheet to indicate what changes in names, value assignments and value labels should be made so the data is consistent across rounds. By doing so, you are also creating helpful documentation about your data work. Once rounds are consistently appended, adapt your construction script so it can be used on the complete panel dataset. In addition to preventing inconsistencies and documenting your work, 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 observed data points to measures of abstract concepts, it is important to document exactly how each variable is derived or calculated. Careful documentation is closely linked to the research principles discussed in the first chapter. It makes research decisions transparent, as anyone can read about how you defined each variable in your analysis, and what was the reasoning behind these decisions. By reading the documentation, 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. Ideally, they should also be able reproduce your steps and recreate the constructed variables. Therefore, documentation is an output of construction as relevant as the code and data, and it is good practice for papers to have an accompanying data appendix listing analysis variables and their definitions.

The development of construction documentation is a good opportunity to have a wider discussion with your team about creating protocols for variable definition, which will guarantee that indicators are defined consistently across projects. Whether you do that or not, make sure to have a detailed account of how variables are created. This will be implemented in your code, but you should still add comments to it explaining in human language what you are doing and why. This 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.

The `iecodebook export` subcommand is a good way to ensure

you have easy-to-read documentation. When all your final indicators have been created, you can use the it to list all variables in the dataset in an Excel sheet. You can then add the variable definitions to that file to create a concise metadata document. Take this opportunity to review your notes and make sure that your code is implementing exactly what is described in the documentation.

Writing data analysis code

After data is cleaned and indicators are constructed, you are ready to generate analytical outputs. There are many existing resources for data analysis and statistical methods, such as *R for Data Science*;[8,9] *A Practical Introduction to Stata*;[10,11] *Mostly Harmless Econometrics*;[12] and *Causal Inference: The Mixtape*.[13] We focus on how to structure analytical code and files, rather than how to conduct specific analyses.

[8] Wickham, H. and Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 1st edition

[9] Available at https://r4ds.had.co.nz

[10] McGovern, M. E. (2012). A Practical Introduction to Stata. PGDA Working Papers 9412, Program on the Global Demography of Aging

[11] Available at https://scholar.harvard.edu/files/mcgovern/files/practical_introduction_to_stata.pdf

[12] Angrist, J. D. and Pischke, J.-S. (2008). *Mostly harmless econometrics: An empiricist's companion*. Princeton university press

[13] https://scunning.com/mixtape.html

Organizing analytical code

The analysis stage usually starts with a process we call **exploratory data analysis**. This is when you are first looking for patterns in your data, creating descriptive graphs and tables, and trying different tests to understand your results. It progresses into **final analysis** when your team starts to decide which are the "main results", or those that will make it into a 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. While this is fine when you are writing your research stream of consciousness into code, it leads to poor practices in the final code such as not clearing the workspace and not loading a fresh constructed dataset before each analysis task.

To avoid mistakes, it's important to take the time to organize the code that you want to keep, that is, the final analysis code, in a clean manner. The result is a curated set of polished scripts that will be part of a reproducibility package. A well-organized analysis script starts with a completely fresh workspace and, for each output it creates, explicitly loads data before analyzing it. This setup encourages data manipulation to be done earlier in the workflow (that is, in separate cleaning and construction scripts). It also prevents you from accidentally writing pieces of analysis code that depend on one another or that require manual instructions for all necessary chunks of code to be run in the right order. IT also encourages you to rewrite each task so it is completely independent of all other code, except for the master script. You could go as far as coding every output in a

separate script.

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 concepts, not the coding. Research questions and statistical decisions should be incorporated explicitly in the code through comments, and their implementation should be easy to detect from the way the code is written. This includes clustering, sampling, and controlling for different 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 functions and 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. It also makes your code more dynamic, as it's easy to update specifications and results through a master file without changing every script.

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 script files to correspond to exhibits as they appear in a paper or report, leave this to near publication time.

Visualizing data

Data visualization[14] is increasingly popular, and is becoming a field in its own right.[15] 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".[16] *Fundamentals of Data Visualization* provides extensive details on practical application;[17] as does *Data Visualization: A Practical Introduction*.[18] Graphics tools in 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.[19] 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.[20] If you are an R user, the *R Graphics Cookbook*[21] is a great resource for the its popular visualization package `ggplot`[22]. But there are a variety of other visualization packages, such as `highcharter`,[23] `r2d3`,[24] `leaflet`,[25] and `plotly`,[26] to name a few. We have no intention

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

[15] 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

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

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

[18] http://socvis.co

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

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

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

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

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

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

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

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

of creating an exhaustive list, but these are good places 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 plotting commands is getting the data into the right format. Though both Stata and R have plotting functions that graph summaries of the data, a good rule of thumb that you can be sure will always work is for each observations in your data set to correspond to one data point your want to represent in your plot. This may seem simple, but may require you to do some of the aggregation and reshaping operations discussed earlier in this chapter.

Based on DIME's accumulated experience creating visualizations for impact evaluations, our team has developed a few resources to facilitate this workflow. First of all, we maintain a **Stata Visual Library**[27], which has examples of graphs created in Stata and curated by us.[28] 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. The `ietoolkit` package also contains two commands to automate common impact evaluation graphs: `iegraph` plots the values of coefficients for treatment dummies, and `iekdensity` displays the distribution of an outcome variable across groups and adds the treatment effect as a note.

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

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

Creating analysis outputs

A great number of outputs will be created during the course of a project. These will include both raw outputs such as tables and graphs and final products such as presentations, papers and reports. During exploratory analysis, your team will consider different approaches to answer research questions and present answers. Though it is best to be transparent about different specifications tried and tests performed, only a few will be considered "main results". When the first outputs are being created, agree on where to store them, what software and formats to use, and how to keep track of them. This discussion will save you time and efforts on two fronts: formatting and polishing tables and graphs that will not make their way into final research products; and remembering the different paths your team has already taken, so you don't do the same thing twice. This section will take you through key elements to keep in mind when making workflow decisions and outputting results.

Managing outputs

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 plain text files, and sync software such as Dropbox and OneDrive 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 plain text formats like `.tex` and `.csv` can be created from most analytical software and managed with Git. Tracking plain text outputs with Git facilitate the identification of changes in 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, so will be automatically flagged for you or a reviewer to check. Tracking changes to binary files such as images is a bit more cumbersome, as they use more space. This may cause your syncing to the cloud to be slower. But it is still a good way to know when changes in the code affect outputs, depending on the git client you are using. GitHub desktop, for example, displays changes in images files in an accessible manner.

Independently of your storage decisions, you will need to update your outputs frequently. And if you have tried to recreate a result after a few months, you probably know that it can be hard to remember where the code that created it was saved. Naming conventions and code organization, including easily searchable file names and comments, play a key role in not re-writing scripts again and again. It is common for teams to maintain one "final" analysis file and one folder with draft code or exploratory analysis. The latter contains pieces of code that are stored for reference, but not cleaned up to be included in any final outputs. Once an output presents a result in the clearest manner possible, it should be renamed and moved to the "final analysis" folder. It's typically desirable to have the names of outputs and scripts linked – so, for example, `factor-analysis.do` creates `factor-analysis.eps` and so on. Document output creation in the master script that runs your code, 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.

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, and 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. Knowing how your code outputs will be used will help you decide

the best format to export them. You can typically use the same command to save figures into different formats, such as `.eps`, `.png`, `.pdf` or `.jpg`. However, the decision between using Office Suite software such as Word and Power Point versus LaTeX and other plain text formats may influence how you write your code, as this choice often implicates in the use of a particular command. We strongly recommend that you chose software to create final products that can be linked to raw outputs in such a way that they are updated in the paper or presentation every time changes are made to them. We broadly call files that have this feature **dynamic documents**, and they will be discussed in more detail in the final section of this chapter.

Exporting analysis outputs

As briefly discussed in the previous section, you do not *need* to export each and every table and graph created during exploratory analysis. Most statistical software allow you to review results interactively, and this is often preferred at this stage. Final analysis scripts, on the other hand, must export outputs that are ready to be included in a paper or report. No manual edits, including formatting, should be necessary after exporting final outputs. Manual edits are difficult to replicate. You may think that it's not worth coding a small formatting adjustment, but you will inevitably need to make changes to the output, and automating them will save you time by the end of the process. On the other hand, don't spend much time formatting tables and graphs until you have come to a decision about which will be used for your final product.[29] Polishing final outputs can be a time-consuming process, and you want to do it as few times as possible.

We cannot stress this enough: do not 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. 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 of a result in a paper or report.

There are numerous commands to export outputs from both R and Stata to a myriad of formats. Some examples are `estout`,[30,31] `outreg2`,[32] and `outwrite`[33] in Stata, and `stargazer`[34] and `ggsave`[35] in R. 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 by Stata. Some publications re-

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

[30] Jann, B. (2005). Making regression tables from stored estimates. *Stata Journal*, 5(3):288–308(21); and Jann, B. (2007). Making regression tables simplified. *Stata Journal*, 7(2):227–244(18)

[31] Documentation available at http://repec.sowi.unibe.ch/stata/estout/.

[32] https://www.princeton.edu/~otorres/Outreg2.pdf

[33] https://www.benjaminbdaniels.com/stata-code/outwrite

[34] https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdf

[35] https://ggplot2.tidyverse.org/reference/ggsave.html

quire "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 fewer. Given our recommendation to use **dynamic documents**, which will be discussed in more detail both in the next section and in chapter , exporting tables to `.tex` is preferred. Excel `.xlsx` and `.csv` are also commonly used, but require the extra step of copying the tables into the final output. The `ietoolkit` package includes two commands to export formatted tables, automating the creation of common outputs and saving time for research. `iebaltab`[36] creates and exports balance tables to Excel or LaTeX. `ieddtab`[37] does the same for difference-in-differences regressions.

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

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

If you need to create a table with a very specific format that is not automated by any command you know, consider writing it manually (Stata's `filewrite` and R's `cat()`, for example, allow 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. 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 included in your code and comments that are not otherwise visible in the output. Examples of this include sample, unit of observation, unit of measurement, and variable definition.[38] It is usually preferable to add labels and notes to tables and figures directly in the documents, rather than through code. This will make their formatting more consistent.

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

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 is not possible in tools like Microsoft Office, although there are tools and add-ons that produce similar functionality. In Word, by default, you have to copy and paste each object individually whenever tables, graphs, or other inputs have to be updated. This workflow becomes more complex as the number of inputs grows, increasing the likelihood of making mistakes or missing updates. Dynamic documents prevent this from happening by managing document compilation and inclusion of inputs in a single integrated process, so you can skip the copying

and pasting altogether. Alternatively, if your team prefers to use static documents they are already familiar with, prefer not compiling outputs into a document at all, at least in exploratory work.

Dynamic exploratory analysis

If all team members working on a dynamic document are comfortable using the same statistical software, built-in dynamic document engines are also a good option. This means you can write both text (often in Markdown[39]) and code in the script, and the result will usually be a PDF or HTML file including code, text, and outputs. In our experience, however, many researchers find the entry cost to learning how to use these tools too high. These types of dynamic document tools are typically best used by the team members that work more closely with code, and can be great for creating exploratory analysis reports as you work on them, or paper appendices including large chunks of code and dynamically created graphs and tables. RMarkdown[40] is the most widely adopted solution in R. Stata offers a built-in package for dynamic documents, `dyndoc`[41], and user-written commands such as `markstat`,[42] `webdoc`,[43] and `texdoc`.[44] The great advantage of these tools in comparison with LaTeX is that they create full documents from within your scripts, so running the code and compiling the document is reduced to a single step.

Documents called "notebooks" (such as Jupyter[45]) 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 for those who are not familiar with them to edit the content, and they often don't offer as extensive formatting options as, for example, Word. 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 can be linked 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 they do offer extensive collaboration features, and can be useful for working on informal outputs. Markdown files on GitHub can also provide similar functionality through the browser, and are version controlled. However, as with other Markdown options, the need to learn a new syntax may discourage take up among team members who don't work with GitHub more extensively.

[39] https://www.markdownguide.org/

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

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

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

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

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

[45] https://jupyter.org

Dynamic research outputs: the case for LaTeX

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.,[46] LaTeX is a document preparation and typesetting system with a unique code syntax. While this tool has a significant learning curve, its enormous flexibility in terms of operation, collaboration, output formatting, and styling make it our preferred choice for most large technical outputs. In fact, LaTeX operates behind-the-scenes in many other dynamic document tools (discussed below). Therefore, we recommend that you learn LaTeX as soon as you are able to.

The main advantage of using LaTeX is that it updates outputs every time the document is compiled, while still allowing for text to be added with a relative simple syntax and user-friendly interface. Additionally, because of its popularity in the academic community, social scientists are more familiar with it than other dynamic documents tools, so the cost of entry for a team is often relatively lower. Because `.tex` files are plain text, they can be version-controlled 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 and include figures and tables that may be updated. It is good for adding small chunks of code into an output. This book, for example, was written in LaTeX and managed on GitHub.[47] Finally, some journals make LaTeX templates available, so papers can be more easily be formatted into their specific layout.

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

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

Chapter 7: Publishing research outputs

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. Similarly, simultaneous collaboration should not involve the repetitive and error-prone task of manually resolving sets of tracked-changes documents with conflicting edits. Furthermore, for most research projects, completing a manuscript is not the end of the task. Academic journals often require a "reproducibility 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 as written work is a new practice for many researchers.

In this chapter, we suggest tools and workflows for efficiently managing collaboration and ensuring reproducibility. First, we discuss how to use dynamic documents to collaborate on technical writing. Second, we provide guidelines for preparing a functioning and informative reproducibility package. If you have organized your analytical process according to the general principles outlined in earlier chapters, preparing to publish materials will not require substantial reorganization of the work you have already done. Hence, publication is 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 written results. In all cases, we note that technology is rapidly evolving and that specific tools noted here may not remain cutting-edge, but the core principles involved in publication and transparency will endure.

Preparing written documents for publication

Development 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. As we outlined in the previous chapter, **dynamic documents** are a way to simplify writing workflows: updates to analytical outputs that appear in these documents, such as tables and figures,

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

can be passed in to the final output with a single process, rather than copy-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 manuscripts as LaTeX documents

As we discussed in Chapter 6, the most widely used software for dynamically managing academic manuscripts is LaTeX (pronounced "lah-tek"). It requires explicitly encoded references to the latest versions of all outputs. This is not possible by default in Microsoft Word. There, you have to copy and paste each object whenever tables, graphs, or other inputs are updated. As time goes on, it becomes more and more likely that a mistake will be made or something will be missed. In LaTeX, instead of writing in a "what-you-see-is-what-you-get" mode as you do in Word, you write plain text in a `.tex` file, interlaced with coded instructions for formatting and exhibits (similar to HTML). LaTeX 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 footnotes and endnotes, mathematical notation, and 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.

To create a LaTeX manuscript, you will typically create a new top-level directory for each final output. Many projects go so far as to create documents like the main text and any appendix text as separate projects. For a single output, one convention is to put the manuscript files (such as `main.tex` and `bibliography.bib`) in the top-level folder, and then have the folders for figures, tables, and other inputs. Your analytical scripts should easily be able to produce outputs at the location of your choice by adjusting paths in the master script; output them here when ready. Above all, keep your workflow as simple as possible. While LaTeX *can* produce complex formatting, this is rarely needed for academic publishing, as your manuscript will usually be reformatted based on the style of the publisher. So it's rarely worth the investment to go beyond basic LaTeX tools: the title page, manuscript sections and subsections, figures and tables, mathematical equations, bolding and italics, footnotes and endnotes, and, last but not least, references and citations.

One of the most important tools available in LaTeX is the BibTeX citation and bibliography manager.[2] BibTeX keeps all the references you might use in an auxiliary `.bib` file, then references them using a

[2]

simple command typed directly in the document. Specifically, LaTeX inserts references in text using the `\cite{}` command. Once this is written, LaTeX automatically pulls all the citations into text and creates a complete bibliography based on the citations you used whenever you compile the document. The system allows you to specify exactly how references should be displayed in text (such as superscripts, inline references, etc.) as well as how the bibliography should be styled and in what order (such as Chicago, MLA, Harvard, or other common styles). 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. BibTeX 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 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 this tool, you can ensure that references are handled in a format you can manage and control.[3] By changing some settings in the document itself, you can change the style of inline citations and bibliographical references. You can, for example, use superscripts or author names inline, and order the bibiliography either by order of appearance or alphabetically. All the formatting and numbering will be handled automatically. You can also choose to display only the references which are cited in the text, or to include every reference in the `.bib` file. Since different publications have different requirements, the ability to adapt this and other formatting very quickly, including through using publisher-supplied templates where available.

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

LaTeX has one more useful trick: using **pandoc**,[4] you can translate the raw document into Word (or a number of other formats) by running the following code from the command line:

[4] https://pandoc.org

pandoc.sh

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

Conversion to Word is still required for a number of publishers. You still don't want to write in Word, though. The `pandoc` utility can only handle relatively simple formatting, but it will seamlessly translate all the features we mentioned before into the correct native Word styling. The bracketed portion after `csl=` specifies the bibliography style; this must be set manually for this conversion to work. You can download a CSL (Citation Styles Library) file[5] for nearly any journal and have it applied automatically in this process. Therefore, even in the case where you are required to provide `.docx` versions of materials to others, or tracked-changes versions, you can create them effortlessly from a LaTeX manuscript, then use external tools like Word's compare feature to generate outputs like integrated track-changes versions when needed.

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

Getting started with LaTeX as a team

Getting used to LaTeX can be challenging, but the control it offers over the writing process is invaluable. Because it is written in a plain text file format, `.tex` can be version-controlled using Git. This makes it possible to set up a dedicated GitHub repository for each output and manage contributions and version histories using the same system we recommend for analytical work. DIME Analytics has created a variety of templates and resources that you can adapt for your own team.[6] Integrated editing and compiling tools like TeXStudio[7] and `atom-latex`[8] offer the most flexibility to work with LaTeX in teams.

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

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

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

Unfortunately, despite these advantages, setting up LaTeX is not always simple, particularly if you are new to working with plain text code and file management. This is because LaTeX requires that all formatting be done in its special code language, and it is not particularly informative when you do something wrong. This can be off-putting very quickly for people who simply want to get to writing, like senior researchers. They can require a lot of troubleshooting at a basic level at first, and staff not used to programming may not easily acquire the necessary knowledge.

In order to make it as easy as possible for your team to use LaTeX without all members having to invest in new skills, we suggest

using a cloud-based implementation such as Overleaf[9] as your first foray into LaTeX writing. Some teams even adopt these tools as a permanent solution. 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.

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

Cloud-based implementations of LaTeX have several advantageous features for teams compared to classic desktop installations. First, since they are completely hosted online, they avoid the inevitable troubleshooting required to set up a LaTeX installation 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 LaTeX coding. Cloud services also usually offer a convenient selection of templates so it is easy to start up a project and see results right away without needing to know a lot of the code that controls document formatting.

Cloud-based implementations of LaTeX also 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. They vary dramatically in their ability to integrate with file systems where you store your code and outputs, and so you will need to practice an integrated workflow depending what is available to you. 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 and publishing data

While we have focused so far on written materials, it is also important to consider how you will publish the data you used for your research. The open science community at large sees data publication both as a citable output and as a necessary transparency measure.

Fortunately, it is a conceptually simple task for you to produce and archive or catalog the required materials. Beginning from the raw data that you collected, you should be prepared to catalog two separate collections. First, you should catalog the unaltered (de-identified) data with all variables corresponding directly to fields in the original dataset or data collection instrument. If you follow the steps outlined in Chapter 5, when you get to the publication stage you will have a cleaned data set and supporting documentation ready. If you follow the steps outlined in Chapter 5, when you get to the publication stage you will have a cleaned data set and supporting documentation ready. Second, you should separately catalog (as supporting material) the analytical dataset used for the output you are publishing. This release should include the data construction scripts that create transformed and derived indicators, as well as project-specific information such as treatment assignment and other indicators that were not directly acquired during fieldwork. Here, following the recommendations from Chapter 6 will ensure that this material is at hand.

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. If you are following the steps outlined in this book, you should have already removed direct identifiers after collecting the data. Direct identifiers are data elements that are keyed to the identity of a respondent, such as names, phone numbers, official identification numbers like government IDs. They further include all information that may easily be linked to public records, such as dates of birth, job titles, marital records, and other personal vital statistics.

At this stage, however, you should additionally remove indirect identifiers, and assess the statistical disclosure risk of your data.[10] Unlike direct identifiers, for which a link (or lack thereof) to public information is verifiable, indirect identifiers require an assessment of the likelihood that an individual can be singled out in the data and then linked to public information in complex combinations. For example, with sufficiently small population groups like US ZIP codes, the age and gender of a respondent may be enough to identify them. This will also often be the case in development work, where information such as the size of a household, the ages and marital statuses of the household members, and the types of work or schooling they engage in may be more than enough to identify a person or family from a sufficiently small group.

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

It also requires an assessment of the potential risk to the individual that could be caused by disclosure of their identity or personal information. This will vary widely depending on the types of information you are collecting and the overall vulnerability of the population. In extreme cases, where the population is highly vulnerable and combinations of information are highly specific, you may not be able to publicly release any data at all. You will still be expected to catalog and cite your data, even if you will not be expected to archive it outside your approved internal system. 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, as discussed in Chapter 5. Out of those, the World Bank's `sdcMicro` tool,[11] in particular, has a feature that allows you to assess the uniqueness of the records in your data. It produces simple measures of the identifiability of records from the combination of potentially indirectly identifying variables, and allows you to apply common information masking algorithms, such as binning, top-coding, and jittering data prior to release. You should determine how sensitive your results are to these transformations if you need to release data from which your exact results cannot be reproduced for this reason.

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

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.[12] 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, 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.

[12] 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

In cases where confidential data is required for analysis, we recommend embargoing sensitive or access-restricted variables when publishing the dataset. In practice, if you have a secure repository from your home institution, funding institution, or other collaborator, you can store the data there and publish only a catalog entry indicating that the data is not available for general publication. 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 raw and constructed datasets

Publicly documenting all original data generated as part of a research project is an important contribution in its own right. Cataloging and/or archiving original datasets is a significant contribution that can be made in addition to any publication of analysis results.[13] 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 catalog entries 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 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. In all cases, you should choose a method that allows you to obtain a digital object identifier (DOI) and a formal citation for your data to use in your publication.

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

Publicly releasing 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 researchers to explore variables and questions that you do not have time to focus on otherwise. There are different options for public data release. The World Bank's Development Data Hub[14] includes a Microdata Catalog[15] and a Geospatial Catalog, where researchers can publish data and documentation for their projects.[16] The Harvard Dataverse[17] publishes both data and code. The Datahub for Field Experiments in Economics and Public Policy[18] 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.[19]

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

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

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

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

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

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

Raw de-identified data can and should be published as an independent entity with its own license. It should be named according to the data collection effort or the project that supported its collection, rather than tied to a specific research output. This makes it easy for you or others to access and cite that dataset. However, even when your raw data is owned by someone else, or for any other reason you are not able to publicly release it, you will often still retain the rights to release derived and constructed data, even if they contain only

the indicators you created and their documentation.[20] Regardless of what may be done with the raw data, you will therefore typically be able to separately license or publish the derived dataset and the indicators needed for analysis as citable supporting materials for any publication. Many of the resources above will support this type of release. The OSF offers a useful functionality to self-publish materials of this form, and some publishers (such as the AEA) offer in-house handling of these materials. If you have questions about your rights over original or derived materials, check with the legal team at your organization or with the data provider. Make sure you understand the rights associated with any data release and communicate them to any future users of the data.

When you do publish data that you own, you will decide how it may be used and what license, if any, you will assign to it.[21] In all cases, material without a license may never be reused; you should prefer to offer a license that is explicit and details whether and how specific individuals may access the data. Terms of use available in the World Bank Microdata Catalog include, in order of increasing restrictiveness: open access, direct access, and licensed access.[22] Open Access data is freely available to anyone, and simply requires attribution. Direct Access data is available to registered users who agree to use the data for statistical and scientific research purposes only, to cite the data appropriately, and not to attempt to identify respondents or data providers or link the data to other datasets that could allow for re-identification. Licensed access data is restricted to bona fide users, who submit a documented application detailing how they will use the data and then sign a formal 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.

Published data should be released in a widely recognized format. While software-specific datasets are acceptable accompaniments to the code (since those precise materials are probably necessary), you should also consider releasing datasets in plain text formats such as CSV files with accompanying codebooks, since these can be used by any researcher. Additionally, you should also release PDF or code versions of 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. With your analytical dataset, you should also release the code that constructs any derived measures from the raw dataset, so that others can learn from your work and adapt it as they like.

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

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

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

Preparing a complete reproducibility package

Major journals now often require that, in addition to the data required to obtain your results, you provide the code that exactly recreates those results. Some even require being able to reproduce the results themselves before they will approve a paper for publication.[23] This set of materials, taken together, is often referred to as a **reproducibility package**. If your material has been well-structured throughout the analytical process, this will only require a small amount of extra work; if not, creating this package may take some time. When it is completed, whoever downloads it should be able to understand how your code produces results from your data and be able to reproduce them by executing the included master script.

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

Organizing code for reproducibility

Before releasing 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. They should also be able to make small adjustments to your code and reproduce individual portions of your analysis so they can understand the choices and functions you implemented. The code should both be functional and readable, through the use of clean structure and extensive commenting and documentation. 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. Whereas your data should already be relatively clean by this stage, your code is much less likely to be so. This is often where you need to invest time prior to releasing your reproducibility package.

DIME has adopted several practices and requirements to support the production of high-quality reproducibility packages by its researchers. The materials for these practices are publicly available, so you can use them to check the reproducibility of your own work. DIME code must be submitted for a computational reproducibility check before publication as a working paper or submission to a journal. This reproducibility check is initiated by contacting the DIME Analytics team and submitting the Reproducibility Package Checklist. In addition to this formal requirement for reproducibility review, DIME projects are required to be maintained continuously at high quality. These practices must include ensuring that project code is organized with a master script so that computational reproducibility is a one-click exercise. DIME projects are also expected to use Git and

GitHub to document project work and collaboration, and to keep the master branch up-to-date as a working edition. Research assistants also participate in quarterly peer code review rounds so that they can improve their code and documentation as it is written, and not be required to revisit it in a rush near publication time.

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 LaTeX document can support this.) Code and outputs which are not used should be removed before publication.

Releasing a reproducibility 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 and there are a variety of archives and storage providers that cater to different needs. 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.

Unlike data, code usually has few external constraints to publica-

tion. 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. You may be required to put your materials into the public domain; if not, it is common to only require attribution and citation for their reuse, without putting any barriers or restrictions to accessing the code.

One option for releasing a reproducibility package is GitHub. Making a public GitHub repository is completely free. It can hold any file types, provide a structured, compressed 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 of 100GB 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. It is not an archive, meaning that it does not guarantee the permanence of uploaded materials, it does not guarantee the permanence of the access URL, and it does not manage citations or non-code licenses by default.

Features to look for in a platform to release reproducibility packages 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. One suggestion is to combine GitHub with the Open Science Framework,[24] as OSF can easily link to and import material from GitHub and apply a permanent URL, DOI, formal citation, general license, and archival services to it.

[24] https://osf.io

Other options include the Harvard Dataverse[25] and ResearchGate.[26] Any of these archival services is acceptable – the main requirement is that the system can handle the structured directory that you are submitting, and that it can provide a stable, structured URL for your project and report exactly what, if any, modifications you have made since initial publication. You can even combine more than one tool if you prefer, as long as they clearly reference each other. For example, one could publish code and the corresponding license on GitHub and point to data published on the World Bank Microdata Catalog. Emerging technologies such as the "containerization" approach of CodeOcean[27] offer to store both code and data in one repository, and also provide an online workspace in which others can execute and modify your code without having to download your

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

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

[27] https://codeocean.com

tools and match your local environment when packages and other underlying software may have changed since publication. In a similar spirit, attempts to create "metadata archives" that link all the different materials, releases, and publications relating to specific research projects or studies offer an avenue for improving the organization of these materials in the future.

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 publicly released before its formal publication date, although, in most development research fields, the release of working papers is a fairly common practice. This can be done on a number of preprint websites, many of which are topic-specific.[28] You can also use GitHub or OSF and link to the PDF file directly through your personal website or whatever medium you are sharing the preprint. However, 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.

[28] https://arxiv.org/

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]

[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

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.

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

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.

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

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. D. and Pischke, J.-S. (2008). *Mostly harmless econometrics: An empiricist's companion*. Princeton university press.

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. (2017). The econometrics of randomized experiments. *Handbook of Economic Field Experiments*, 1:73–140.

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.

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

Christensen, G., Freese, J., and Miguel, E. (2019). *Transparent and reproducible social science research: How to do open science*. University of California Press.

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.

Duflo, E., Banerjee, A., Finkelstein, A., Katz, L. F., Olken, B. A., and Sautmann, A. (2020). In praise of moderation: Suggestions for the scope and use of pre-analysis plans for rcts in economics. Technical report, National Bureau of Economic Research.

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.

Gelman, A. and Loken, E. (2013). The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University*.

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.

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

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*.

Jann, B. (2005). Making regression tables from stored estimates. *Stata Journal*, 5(3):288–308(21).

Jann, B. (2007). Making regression tables simplified. *Stata Journal*, 7(2):227–244(18).

Kerr, N. L. (1998). Harking: Hypothesizing after the results are known. *Personality and Social Psychology Review*, 2(3):196–217.

McGovern, M. E. (2012). A Practical Introduction to Stata. PGDA Working Papers 9412, Program on the Global Demography of Aging.

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.

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., and Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606.

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.

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*.

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. and Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 1st edition.

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

Index

LaTeX, 104
`DataWork` folder, 32
`iefolder`, 32
`ietoolkit`, 32
LaTeX, 33, 102

personally-identifying information, 81

attrition, 65

backup, 28

clustered randomization, 55
code organization, 34
code review, 36
collaboration tools, 29
contamination, 65
counterfactual, 45
cross-sectional data, 47

data analysis, 95
data collection, 39
data maps, 44
data storage, 41, 42
data transfer, 41, 42
data visualization, 96
de-identification, 41, 82
Documentation, 87
Dropbox, 28
dynamic documents, 33

email, 29
encryption, 40–42, 69
experimental research designs, 46

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

geodata, 41
GitHub, 21

human subjects, 39, 81

iecodebook, 87
iefieldkit, 76
Institutional Review Board, 39
instrumental variables, 47

longitudinal data, 48

Markdown, 33
master datasets, 44
master do-file, 35
matching, 47
monitoring data, 48

Open Science Framework, 21

panel data, 48
password protection, 28
personally-identifying information, 39
power calculations, 56

Pre-analysis plan, 89, 90
pre-analysis plan, 19
pre-registration, 18
privacy, 41
project documentation, 21

quasi-experimental research designs, 46
questionnaire design, 64

randomization inference, 57
randomized control trials, 46
regression discontinuity, 46
repeated cross-sectional data, 48
Research design, 89

sampling, 50
server storage, 29
software environments, 30
software versions, 31
statistical disclosure, 41, 108
stratification, 55

task management, 21

version control, 29

WhatsApp, 29