SOC 4650/5650 User's Guide

Christopher Prener, Ph.D.

2017-01-08

Contents

Preface

This text is a companion document for **SOC 4650/5650 - Introduction to Geographic Information Sciences**. It is designed to help you be *successful* in this course. The idea behind a course **User's Guide** is to create a reference for many of the intangible, subtle or disparate skills and ideas that contribute to being a successful researcher. In creating a **User's Guide**, I draw inspiration from the work of Donald Knuth.[1] Knuth has discussed his experiences in designing new software languages, nothing that the developer of a new language

> …must not only be the implementer and the first large-scale user; the designer should also write the first user manual… If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important…

While there is nothing particularly new about what I am writing here, and I am certainly not developing a new language for computing, the goal of the **User's Guide** remains similar to Knuth's experience. By distilling some of key elements for making a successful transition to being a *professional developer* of knowledge rather than a *casual consumer*, I hope to both improve the course experience itself and also create an environment that fosters a successful learning experience for you.

If you read through the course objectives included in the syllabus, you will note that creating maps is only one of them. As much as this is a GIS course, it is a course in research methods. In particular, we are concerned with *high quality* research methods and the *process* of conducting research. We therefore focus on a combination of mental habits and technical practices that make you a successful researcher. Some of the skills and techniques that we will discuss this semester are not taught as often in graduate programs. Instead, they are often the products of "learning the hard way". These "habits of mind and habits of method" are broadly applicable across methodologies and disciplines.

License

[1] Donald Knuth is the developer of TeX, a computer typesetting system that is widely used today for scientific publishing in the form of LaTeX. He also established the concept of literatue programming, which forms the basis of some of the practices we will follow with Stata this semester.

Chapter 1

Getting Started

Before you begin the semester, there are a number of things that I recommend that you do to help set yourself up for success. Before you do *anything* else, you should read through the **Syllabus** and the **Reading List**. Make sure you have a good sense of what is *required* for the course. If you have questions, bring them to the first day of class!

1.1 Prep Your Computer

Before you do anything else for this course, make sure you get your computer ready for the work you are about to undertake:

1. Make sure your operating system is up-to-date. If you are able, I would also recommend upgrading your computer to the most recent release of its operating system that the computer can run.
2. We'll be sharing computer files throughout the semester, so you should ensure that you have functioning anti-virus software and that it is up-to-date.
3. You'll also need to download files, so you'll need to make sure you have some free space on your hard drive. If you have less than 10GB of free space, you should de-clutter!
4. Make sure you know how to access your computer's file management system.
    - On macOS, this means being comfortable with Finder.app.
    - On Windows, this means being comfortable with Windows Explorer.

This of course assumes that you own a computer. Owning a computer is not required for this course. All students who are enrolled in SOC 4650 or SOC 5650 will be given 24-hour swipe access (*just what you always wanted!*) to Morrissey Hall to facilitate access to lab computers.

1.2 Create Accounts

There are two major web services that we will use this semester, and you'll need to create accounts for both:

- **GitHub** - you can sign-up at GitHub.com. Once you've signed up, fill out your profile, set-up two-factor authentication, and let Chris know (via email) what your user name is. Once he has it, he can add you to the SOC 4650/5650 organization.
- **Slack** - you can ask Chris (via email) for an invitation to sign-up for our team. Once the sign-up process is complete, you can log-in by going to our team's Slack site. Fill out your profile, set-up two-factor authentication, and change your timezone.

1.3 Download and Install Software

There are a number of software applications that we will use this semester. Most of them are free, and I recommend downloading those free ones right away. All of these applications are available for macOS and Windows.

- **Atom** - Atom is a flexible, open-source text editor that is produced by GitHub. You can download it from Atom's website.
- **GitHub Desktop** - GitHub makes a desktop client that you can use to easily interact with repositories that are stored on the site. You can download it from GitHub's website after you sign-up for an account there. You'll need that account information to complete the desktop client's set-up process.
- **Slack** - Slack has a number of applications for desktop and mobile operating systems. I recommend downloading Slack on your personal computer, and optionally installing it on your mobile device as well. You can download their desktop applications from their website and the mobile applications from your App Store.

For Graduate Students *only*

If your computer meets the operating system requirements for ArcGIS and you think you'd benefit from having access to the software at home, let Chris know (via email).

If you are in the Public and Social Policy Ph.D. program and your computer meets the hardware and software requirements for Stata, you should consider purchasing it for yourself. I recommend purchasing a perpetual license for Stata/IC. This is the most cost-effective solution for typical students.

1.4 Buy Course Materials

Books

There are three required books for this course:

1. Brewer, Cynthia. 2015. *Designing Better Maps: A Guide for GIS Users.* Redlands, CA: ESRI Press. ISBN-13: 978-1589484405; List Price: $59.99; ebook versions available.
2. Gorr, Wilpen L. and Kristen S. Kurland. 2013. *GIS Tutorial 1: Basic Workbook.* 10.3.x edition. Redlands, CA: ESRI Press. ISBN-13: 978-1589484566; List Price: $79.99; ebook versions available.
3. Thomas, Christopher and Nancy Humenik-Sappington. 2009. *GIS for Decision Support and Public Policy Making.* Redlands, CA: ESRI Press. ISBN-13: 978-1589482319; List Price: $24.95.

There is one additional book that is optional:

- Mitchell, Michael N. 2010. *Data Management Using Stata: A Practical Handbook.* College Station, TX: Stata Press. ISBN-13: 978-1597180764; List Price: $48.00.

Buying Mitchell (2010) is *highly* recommended for graduate students who will continue using Stata in the future and those who are concerned about the command-line interface. I recommend waiting for a week or two before purchasing this.

External Media

You will need a USB external storage device (either an external hard drive or a thumb-style drive) that has at least 20GB of storage capacity. This will be used for storing spatial data for this course.

1.5 Download Course Data

Mots of the course data is available for download via Dropbox in a single `.zip` file. If you want, you can let Chris know (via email) that you'd like to download these data before the beginning of the semester. Once you download them, extract the data from the `.zip` file and transfer them to your external storage device.

Chapter 2

Approaching this Course

Students have varying experiences learning GIS techniques. For some, the spatial logic and programming that are the foundation for GIS methods come naturally. For others, being introduced to these concepts can be an anxiety producing experience. I am fond the phrase "your mileage will vary" for describing these differences - no two students have the exact same experience taking a methods course.

2.1 Zen and the Art of Data Analysis

One of the biggest challenges with this course can be controlling the anxiety that comes along with learning new skills. ArcGIS processes, Markdown syntax, and Stata commands can seem like foreign alphabets at first. Debugging Stata do-files can be both challenging and a large time suck, in part because you are not yet fluent with this language. Imagine trying to proofread a document written in a language that you only know in a cursory way but where you must find minute inconsistencies like misplaced commas.

For this reason, I also think it is worth reminding you that many students in the social sciences struggle with quantitative methods at first. It is normal to find this challenging and frustrating. I find that students who can recognize when they are beginning to go around in circles are often the most successful at managing the issues that will certainly arise during this course. Recognizing the signs that you are starting to spin your wheels and taking either ten minutes, an hour or two, or a day away from GIS coursework is often a much better approach than trying to power through problems.

2.2 An Apple a Day

Being able to walk away from an assignment for a day requires excellent time management. If you are waiting until the night before or the day of an assignment's due day to begin it, you give yourself little room for errors. I recommend approaching this course in bite size chuncks - a little each day. The most successful students do not do all of their reading, homework, and studying in a single sitting. I find that this approach not only creates unnecessary anxiety around assignments, it also dramatically limits the amount of course material you can absorb. Keep in mind that I expect the *median* student to spend approximately six hours on work for this class each week (twice the amount of in-class time).

A sample approach to the class might look something like this:

- Tuesday: class
- Wednesday: finish lab
- Thursday: Start problem set
- Friday: Finish problem set

- Saturday: First reading
- Monday: Second reading

2.3 Reading with Purpose

The book and article **reading assignments** for this course are different from most of the other reading you will do in your graduate program because they are often very technical. Students who are most successful in this course read twice. Read the first time to expose yourself to the material, then take a break from the reading. During this first read, I don't recommend trying to complete the example problems or programming examples. Focus on the *big picture* - what are the concepts and ideas that these readings introduce?

During the second read, try to focus in in the *details* - what are the technical details behind the big picture concepts? I recommend doing this second read with your computer open. Follow along with the examples and execute as much of them as you can. By using this second read through as a way to test the waters and experiment with the week's content, you can come into the lecture better prepared to take full advantage of the class period. Students who follow this approach are able make important connections and focus on the essential details during lectures because it is their third time being exposed to the course material. They are also in a much stronger position to ask questions.

2.4 Active Lectures and Labs

During **lectures**, I introduce many of the same topics that your readings cover. This again is intentional - it gives you yet another exposure to concepts and techniques that are central to geospatial science. One mistake students sometimes make is focusing on the details of *how* to do a particular task rather than focusing on *when* a task should be done. If you know when a task is needed but cannot remember how to do it in Stata or ArcGIS, you can look this information up. Conversely, detailed notes on executing Stata commands may not be helpful if you are unsure when to use a particular skill. There is no penalty in this course for not knowing how to execute a command from memory; this is what reference materials are for. The most successful students will therefore focus on *when* a particular skill is warranted first before focusing on *how* to execute that skill

Getting experience with executing tasks is the purpose of the **lab exercises**. Time for beginning these exercises is given at the end of each class meeting, and replication files will be posted on GitHub for each lab.

2.5 Typefaces, Fonts, Files, and Examples

2.5.1 Typefaces and Fonts

Stata publications use a `monospaced typewriter style typeface` to refer to Stata commands (inputs) and Stata results (outputs). I take the extra step of highlighting commands with a when they are referenced in a sentence. In some documents, like lecture slides and cheat-sheets, I may highlight a command by using a to increase the visibility of the command name itself.

The `typewriter typeface` is also used to refer to filenames (e.g. `auto.dta`) or filepaths (e.g. `C:\Users\JSmith\Desktop`). Finally, we will use the `typewriter typeface` to refer to GitHub repositories (e.g. `Core-Documents`, the repository that contains this file).

Stata publications use *italicized text* to refer to text that is meant to be replaced. These references will typically appear in a `typewriter typeface` since they are often part of commands. For example, `describe` `varname` (with `varname` *italicized*) indicates that you should replace the text `varname` with the appropriate variable name from your dataset.

Stata publications use a sans serif typeface to refer to areas of the Stata user interface, menu items, and buttons. Stata publications also use a sans serif typeface to refer to keyboard keys (e.g. Crtl+C) where the plus sign (+) indicates that you should press multiple keys at the same time.

A sans serif typeface combined with a right facing triangle-style arrow (>) is used to refer to actions that require clicking through a hierarchy of menus or windows (e.g. File > Save).

FYI, Since you are reading this document rendered as a Markdown file, you can't see exact examples of what a sans serif typeface looks like.

2.5.2 Stata Files

There are a number of file types that are important for our use of Stata. These are all likely file types that you have never come across before, and are all discussed in greater detail in the Introducing Stata chapter (see page).

- `.do` - "Stata Do-files" - These are code files that contain commands that Stata can execute automatically. All final analyses and manipulations for research should be done via do-files to increase project documentation and reproducibility.
- `.dta` - "Stata Datasets" - These are the format that Stata stores tabular data in. We call these "D-T-A" files.
- `.smcl` - "Stata Log-files" - The default file format for Stata log-files is the `.smcl` file format, which is a varient of **`html`**. It is pronounced "smick-el". I recommend avoiding this file format whenever possible since only Stata can read it. Instead, save your log-files using the `.txt` file extension and the `text` option. The `.txt` file-type is a so-called "plain text" file format that can be read by an innumerable number of applications. This makes it excellent for reproducibility.

2.5.3 Other Files

We will also use a number of other types of files throughout the semester. Some may be file types that you have come across before.

- `.md` - "Markdown files" - These are plain text files that contain Markdown syntax (see page). They are saved with a special file extension so that software applications and web browsers know to take advantage of the embedded Markdown syntax.
- `.png` - "Portable Network Graphics" or "PNG files" - These are image files designed primarily for use on the Internet and on computer displays.
- `.txt` - "Plain text files" or "Text files" - These are files that contain text without any formatting (like bold or italicized text, for example). These can be opened by a wide array of text editor applications across all major operating systems.

2.5.4 Examples

Throughout the semester, I will give you examples both in lecture slides and in an example do-file. Examples in lectures and course documents can be easily identified by their use of the `typewriter typeface`:

```
. summarize mpg
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
```

```
         mpg |         74     21.2973    5.785503         12         41
```

Examples will almost always use the file `census.dta`, which comes pre-installed with Stata. To open it, use the `sysuse` command: `sysuse census.dta, clear`. This allows you to easily recreate examples by minimizing dependencies within do-files.

Chapter 3

“Good Enough” Research Practices

This section introduces some of the core concepts that we will emphasize in this course throughout the semester. The title takes inspiration from a recent article titled “Good Enough Practices in Scientific Computing”[1]. The authors note in their introduction that scientific computing advice can sometimes be both overwhelming and focused on tools that are inaccessible to many analysts. Their goal, and the goal of this course, is to de-mystify the simplist tools that enable researchers to streamline their workflows:

> Our intended audience is researchers who are working alone or with a handful of collaborators on projects lasting a few days to a few months, and who are ready to move beyond emailing themselves a spreadsheet named `results-updated-3-revised.xlsx` at the end of the workday…Many of our recommendations are for the benefit of the collaborator every researcher cares about most: their future self.

I would argue that the skills they describe are useful beyond just a few months. Indeed, most of the skills here can dramatically improve students’ dissertation experiences:

> Most importantly, these practices make researchers more productive individually by enabling them to get more done in less time and with less pain. They also accelerate research as a whole by making computational work (which increasingly means all work) more reproducible. But progress will not happen by itself. Universities and funding agencies need to support training for researchers in the use of these tools. Such investment will improve confidence in the results of computational work and allow us to make more rapid progress on important research questions.

While much of what we will talk about in this course is aimed at supporting your work, there are benefits that extend beyond your dissertation or your research projects. These benefits, which include developing sustainable workflows and structuring the way you interact with your own computer, can make everyday computing practices like checking email or organizing files an easier, more structured process.

3.1 Reproducibility

One of the mantras of this course is our emphasis on reproducibility. The unifying feature of all of the “good enough” research practices discussed below is that they contribute to a more reproducible research product.

Reproducibility is very much in vogue right now for number of reasons. Assessments of studies in psychology[2], for example, have found weaker on average effect sizes and far fewer statistically significant results than the initial studies reported. There have also been high profile instances of falsified research, including research

[1] Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L. and Teal, T.K., 2016. Good Enough Practices in Scientific Computing. *arXiv preprint arXiv:1609.00037.*

[2] Open Science Collaboration, 2015. Estimating the reproducibility of psychological science. *Science*, 349(6251), p.aac4716.

by a graduate student at UCLA. This particular instance of fraud was identified by gradute students intent on replicating the original study.

At the same time, there is a recognition that the skills necessary for producing reproducible research are not being fostered in academic disciplines and graduate programs. Thus one of the goals of this course, and this **User's Guide** in particular, is to help develop a working knowledge of many of these skills.

One challenge, however, is that reproducibility does not have a consistent definition. Some researchers use the term to narrowly refer to code that can execute without alteration on a person's computer. Others use it to refer to research designs that can be replicated by other researchers. Still others discuss reproducibility as the ability to obtain a similar set of results or draw similar inferences from identical research designs.

When we talk about reproducibility in this class. We'll be primarily concerned with **methods reproducibility**:

> the ability to implement, as exactly as possible, the experimental and computational procedures, with the same data and tools, to obtain the same results.[3]

Methods reproducibility in GISc means that other analysts have full access to both the original data and the steps used to render those original data into a final research product, such as a set of maps.

For GISc, this ability is derived from a number of sources. The first source is the use of **computer code** for working with data. Rather than making manual changes to tabular data in a spreadsheet application like Microsoft Excel, computer code provides detailed records of each individual alterations. Code can be used execute tasks repeatedly, meaning that errors can be easily fixed if they are discovered an hour, a day, a week, or a month later. During this semester, we'll use Stata's programming language to execute reproducible data cleaning processes.

Operations in ArcGIS can also be scripted using the programming language Python. Python is an open-source language that is widely used by data analysts and computer programmers. We will not learn ArcPy, the library of Python commands for ArcGIS, this semester. However, it is important to know that many of the things we will learn this semester *can* be scripted, dramatically increasing the reproducibility of your work.[4]

Since we won't focus on scripting for ArcGIS this semester, much of the work we will do will be done manually. This means that no record exists of the changes we make or the steps that we take to complete a task. From a reproducibility standpoint, this is problematic. Even if we were scripting our work in ArcGIS, there are often aspects of projects that must be completed manually. In GISc, this often arises in initial steps like download data or in the production of final map products, which often require using graphic design software.

The second source of reproducibility in GISc is therefore derived from the **documentation** that we create to accompany our research products. These documents outline where our data originated (GIS metadata files), what specific variables mean (a codebook), what steps were taken to create specific maps (a research log), and how our data files are organized (a metadictionary).

The third and final primary source of reproducibility in GISc is derived from our **organizational approach** to our work. GISc projects can require many gigabytes of data spread across dozens or even hundreds of files, feature classes, and databases. A disorgnaized file system can make replicating your work difficult if not impossible. Much of the research practices discussed in the remainder of this section are aimed at supporting one or more of these three major sources of reproducibility.

[3] Goodman, S.N., Fanelli, D. and Ioannidis, J.P., 2016. What does research reproducibility mean?. *Science translational medicine*, 8(341), pp.341ps12-341ps12.

[4] For those of you who are interested, we'll be providing Python/ArcPy examples for many of the ArcGIS taks we learn this semester. These will be available on GitHub in the `ArcPy` repository for those of you interested in expanding your knowledge.

3.2 Thinking in Workflows

One way to increase the reproducibility of a project is to approach each and every task with purposeful organization and thoughfulness. **Workflows** are the processes that we use to approach a given task. Think of checking your email. You (hopefully!) follow a series of steps when you check your email that help you organize your inbox. When I check my email for the first time each day, my workflow looks something like this:

1. Delete junk mail
2. Read and then delete New York Times and Washington Post morning newsletters
3. Read and then delete SLU Newslink newsletter
4. For each remaining email:
    a. Respond if response will take less than two minutes and/or
    b. forward to task management inbox if email requires an action, or
    c. snooze[5] the message until "later today" or "tomorrow morning" if response will require more time than currently avaiable.

In our reading for the first week of classes, Scott Long[6] describes a structured strategy for approaching statistical research. In Long's model, a data analysis project consists of four steps: (a) data cleaning, (b) analysis, (c) presenting results, and (d) protecting files. This is a useful model to build upon for GISc work, and one that we will discuss over the course of the semester.

Even more useful, not just for GISc work but for any process, are the tasks Long lays out for each step in the data analysis workflow:

1. Planning
2. Organization
3. Documentation
4. Execution

A good example of the utility of extending this logic to other workflows is with the problem sets. The "typical" approach students take with homework assignments is to sit down, open up their software, and start with question 1. Using Long's four task approach, a workflow-based strategy to the assignment would involve beginning by reading the assignment through in its entirety to develop a **plan** for approaching it - think about what techniques and skills are needed for each step. With a plan in place, you can proceed to **organizing** yourself for the assignment - identifying and obtaining files that you will need, creating dedicated directories for saving assignment data, and getting any necessary software documentation. After pulling together all of these materials, you are ready to move on to **documentation** - setting up your assignment code and output files, and (later in the course) your research log and meta-dictionary. Once you are set-up, you would then begin to address individual assignment questions as part of the **execution** task.

The goal here is to approach everything you do for research or work with an element of mindfullness and structure about your process. This mental model for approaching research supports the creation of **reproducible** research products because we approach our work in a routinized, predictable, organized, and efficient manner. Thinking in terms of workflows also encourages a greater awareness of the complexity of tasks, which also helps you plan more accurately for how long a particular task or project will take.

In reality, there will be multiple workflows that you find yourself navigating. You will want a structured process not just for approaching a large spaital research project like the final project, but also a process for maintaining notes related to a specific assignment, a process for documenting code, a process for approaching assignments, and even a process for backing your data up. As you go through the course, think about how to best integrate these ideas into your work habits.

[5] Snoozing is a "magical" feature of the email client that I use - SparkMail.

[6] Long, J.S., 2009. *The workflow of data analysis using Stata.* College Station, TX: Stata Press.

3.3 Data Management

One of the themes in "Good Enough Practices in Scientific Computing" is an emphasis on data management. One of their core messages is to "save the raw data". In GISc work, the raw data can be expansize - dozens of shapefiles, tabular files, and associated metadata. These files often come from disparate sources - city open data sites, the U.S. Census Bureau, state data repositories, and other federal agencies. Moreover, GIS data are often updated over time to reflect on-the-ground changes. Saving the raw data in GISc work therefore means not only creating a well-organized directory containing *all* of your original data. It also means logging the source of each file, when it was downloaded, and (if applicable) a permanent web link to your data source.

A second message in the paper is to "create the data you wish to see in the world". The authors encourage readers to "create the dataset you wish you had received." First and foremost, this means using open and not proprietary data formats. For spatial data, ESRI shapefiles are technically proprietary, though their standard is open. This means that other software applications, like `R`, QGIS, and even Stata can read and in some cases write shapefiles. For sharing spatial data, a better option is the GeoJSON, which is a plain text file format.

Tabular data are best stored as `CSV` files, which is also a plain text file format that can be opeend by a wide variety of applications. In contrast, common file formats like Microsoft Excels's `XLS` and `XLSX` are proprietary file packages that cannot be read as plain text and are therefore less desireable for storing data.

Both tabular and spatial data, in their final forms, should be what we consider "tidy data"[7] Tidy data are defined by a number of common attributes - each column represents a single variable or attribute and each row represents a single, unqiue observation. This arrangement should produce clear, easy to read datasets.

Tidy datasets also have other characteristics. Variable names should be short, clear, and self-explanatory (i.e. `streetAddress` and `zipCode` are preferable to `add1` and `add2`). Missing data should be properly declared in a machine-readable format instead of using a code like `-1` or `9999`. Filenames should also be clear and self-explanatory (i.e. `stlouisHomes_011717.csv` is preferable to `final.csv`).

In GISc, we also want to follow another maximum from "Good Enough Practices in Scientific Computing" - anticipate the need for multiple tables. We often have to merge data from multiple sources to produce our final maps, and this requires that the identification variables are all formatted and stored in the same way (i.e. `1234` and `1234`, not `1234` and `1,234`). If identification variables do not match, software applications will not be able to join the multiple tables together.

- Record all the steps used to process data

3.4 Software

- Place a brief explanatory comment at the start of every program
- Make dependencies and requirements explicit
- *follow literatre programming*

3.5 Collaboration

- Create an overview of your project
- Create a shared public "to-do" list
- Make the license explicit

[7] Wickham, H., 2014. Tidy Data. *Journal of Statistical Software*, 59(i10).

3.6 Project Organization

- Borrow from Long
- Put each project in its own directory, which is named after the project
- Put text documents associated with the project in the doc directory
- Put raw data and metadata in a data directory, and files generated during cleanup and analysis in a results directory
- Name all files to reflect their content or function

3.7 Tracking Changes

- Back up (almost) everything created by a human being as soon as it is created
- Keep changes small
- Share changes frequently
- Create, maintain, and use a checklist for saving and sharing changes to the project
- Store each project in a folder that is mirrored off the researcher's working machine

Chapter 4

Protecting Your Work

Each semester that I teach this course or SOC 5050 (Quantitative Analysis), two things happen. The first thing that happens is that students regularly lose files. The effects of losing files can range from being a minor frustration to a major headache depending on the file in question. Losing files often results in downloading multiple copies of the same data and recreating work. Both of these are wastes of your time. Moreover, files are rarely gone. They are typically just misplaced. This is bad for reproducibility, particularly when you happen across multiple versions of the same file and have to sort out which version is the version you last worked on.

The second thing that happens is that students lose their thumb drives. Depending on the timing of this loss, this can again range from being a minor frustration (very early in the semester) to being downright anxiety attack producing (last few weeks of the semester). Recreating an entire semester's worth of work on the final project is both a tremendous waste of your time and a particularly unpleasant experience.

Fortunately, I have never had a student's computer hard drive die during the course of the semester. However, I assume that if I teach this course long enough a hard drive failure will indeed occur. The backup provider Backblaze has analyzed their own hard drives and found that about 5% of drives fail within the first year. After four years, a quarter (25%) of drives in their data center fail.

Similarly, it is only a matter of time before a student's computer is stolen along with all of their hard work. A less likely though still very plausible scenario involves the destruction of a student's belongings (computer and thumb drive included) in a fire, car accident, or natural disaster.

Despite the likelihood that you will at some-point lose a thumb drive (if not during this semester than sometime down the road) and the near certainty that your computer's hard drive will eventually fail if a rogue wave does not get it first, few students and faculty take these risks seriously. While you cannot prevent many of these things from happening, I want to suggest to you that you can take some simple steps to sure that *when* (not if) they happen, you are well prepared to get back to work with minimal disruption.

4.1 Creating a Sustainable File System

In his excellent document *The Plain Person's Guide to Plain Text Social Science*, Kieran Healy describes two important revolutions in computing that are currently taking place. One of them is the advent of mobile touch-screen devices, which he notes

> hide from the user both the workings of the operating system and (especially) the structure of the file system where items are stored and moved around.

For most users, I would argue that this extends to their laptop or desktop computers as well. I would venture to guess that the majority of my students are used to keeping large numbers of files on their desktops or in an (distressingly) disorganized `Documents` folder.

For research, particularly quantitative research, such an approach to file management is unsustainable. It is difficult to produce *any* research, let alone work that is reproducible, without an active approach to file management.

4.1.1 Create a *Single* Course Directory

The most successful approach to organizing files is to identify *one and only one* area that you will store course files in. Having files scattered around you hard drive between you `Desktop` directory, `Downloads`, `Documents`, and a half dozen other places is a recipe for lost files. It can also add complexity to the task of backing these files up. I recommend naming this directory simply `SOC4650` or `SOC5650`. This is short, has no punctuation or spaces (which can create conflicts with software), and explicitly connects the directory to this course as opposed to other courses you may take that are also GIS courses (a good reason to avoid naming the directory `GIS`!).

4.1.2 Approach Organizing Systematically

Within your single course directory, I recommend following much of Long's (2009) advice on organization. Approach this task systematically and mindfully. This approach begins with having a number of dedicated subfolders within your course directory:

```
/SOC5650
  /Core-Documents
  /Data
  /DoeAssignments
  /FinalProject
  /Labs
  /Lectures
  /Notes
  /ProblemSets
  /Readings
  /WeeklyRepos
```

Note again how these directories are named - there are no spaces, special characters, and the names are deliberately short but specific. For a directory with two words (`CoreDocuments` or `FinalProject`), I use what is known as camelCase to name the file where the second (any any subsequent) words have their first character capitalized. You could also use dash-case (`Core-Documents`) or snake_case (`Core_Documents`) as a naming strategy. Regardless of which of these approaches you take, use it consistently.

The course data release is embedded in an otherwise empty folder structure that mirrors this layout. When you download these data and the accompanying directories, un-zip them and move the entire contents to the root of your thumb drive or external hard drive. If you are registered for SOC 4650 and want your directory to match your registration, feel free to rename it `SOC4650`.

4.1.3 The `Core-Documents` Directory

This directory will *not* be included in the folder structure that you download along with the course data release. This directory will be added to your file system during **Lab-01**, when it is **cloned** from GitHub. A cloned directory is one that retains a digitial link to the data stored on GitHub, meaning that it can be easily updated if changes are made. This will be explained in greater depth in the next chapter of the User's Guide. **Do not edit the files in these repositories.**

4.1.4 The Data Directory

The data directory should have copies of all original data and their documentation. Most of these data are included in the initial data release, but you will have to add some additional data to this directory over the course of the semester. The data in this directory should be used as needed but not altered (one of the of the "good enough" research practices from the previous chapter).

4.1.5 The DoeAssignments Directory

Like the `Core-Documents` repository, this will not be included in the course data release. You will add it to your file system during **Lab-01**. It will also have a different name - your last name instead of 'Doe'. Once you add it, it will contain a number of subdirectories:

```
/SOC5650
  /DoeAssignments
    /FinalProject
      /Documentation
      /Memo
      /PosterDraft
      /PosterFinal

    /Labs
      /Lab-01
      ...
      /Lab-16

    /ProblemSets
      /PS-01
      ...
      /PS-10
```

The `FinalProject` directory contains submission folders for each component of the final project. If you a registered for SOC 4650, your directory will look like what appears above. Students registered for SOC 5650 will have three additional subfolders for deliverables related to the final paper element of the course.

The `Labs` and `ProblemSets` directories have subfolders dedicated to the 26 individual assignments you'll have to submit over the course of the semester. **These directories are intended to store only the deliverables that are requested in each assignment's directions.** All other files related to each assignment should be stored elsewhere.

4.1.6 The FinalProject Directory

The final project directory should be a microcosm of the larger directory structure, with most major directories replicated so that your final project files have a dedicated, organized home:

```
/SOC5650
  /FinalProject
    /Data
    /DataAnalysis
    /Documentation
    /Memo
    /Notes
    /Poster
    /Readings
```

You'll notice that there are a number of new directories dedicated to specific aspects of the project.

SOC 5650 students: you will want to add directories for the `/AnnotatedBib` and `/Paper` aspects of the assignment. I also recommend using some type of bibliography software. (Endnote, for example, can be obtained for free by SLU students). Whatever application you choose, keep its primary database for your project in the `Readings` folder along with copies of all `.pdf` readings.

4.1.7 The `Labs` Directory

This directory contains subfolders for each of the sixteen lab assignments for this course. Save *all* of the associated materials for each lab assignment here, including text files, documentation, map files and output, data tables, and any new data that you are asked to create and save.

4.1.8 The `Lectures` Directory

This directory contains subfolders for each of the sixteen weeks of the course. When we create new data files, make maps, or write code during lectures, save these documents in the appropriate week's folder.

4.1.9 The `Notes` Directory

Use this as a home for course notes.

4.1.10 The `ProblemSets` Directory

This directory contains subfolders for each of the ten problem set assignments for this course. Save *all* of the associated materials for each problem set here, including text files, documentation, map files and output, data tables, and any new data that you are asked to create and save.

4.1.11 The `Readings` Directory

Use this as a home for `.pdf` copies of course readings.

4.1.12 The `WeeklyRepos` Directory

Clone each of the weekly repos to this directory, and sync them when updates are made to ensure you have the latest versions of files.

Do not edit the files in these repositories. If you want or need to work with them, make a copy and save it into the relevant assignment directory.

4.2 Backing Up Your Data

There are a number of different ways to think about backing up your data. The most successful backup strategies will incorporate all of these elements.

4.2.1 Bootable Backups

"Bootable" backups are mirrored images of your *entire* hard drive, down to temporary files, icons, and system files. With a bootable backup, you can restore your entire computer in the event of a hard drive failure or a corruption of the operating system files. They are named as such because you can plug in the external drive that you are using for this backup and literally boot your computer up from that drive (typically a *very* slow process).

These backups are often made less frequently because they can be resource intensive and it is best not to use your operating system while creating a clone. They are typically made to an external hard drive, which is subject to similar failure rates as the hard drives inside your computer. So bootable drives need to be replaced every few years to maintain their reliability.

Both major operating systems come with applications for creating clones of your main hard drive that are bootable, and there are a number of third party applications that provide this service as well.

4.2.2 Incremental Backups

Incremental backups are designed to keep multiple copies of a single file (how often depends on the type of software you use and the settings you select). These can be used to restore an older copy of a file if work is lost or a newer file is corrupted.

Apple's TimeMachine is a great example of an incremental backup - when kept on, it creates hourly backups of files that have been changed, daily backups for the previous month, and weekly backups for previous months. Once the disk is full, the oldest backups are deleted. Dropbox also provides a similar service, retaining all previous versions of files (and deleted files) for thirty days.

Incremental backups are typically good options for recovering files that have been recently changed (again, depending on the software you use and the settings you select). Since they run frequently (every time a file is changed or every hour, for example), recent changes tend to get captured. They can be limited in terms of their long-term storage - it may not be possible to recover older versions of a file past a few weeks.

They are also not always good solutions for recreating your entire computer since they do not save all necessary program and operating system files, and may be cumbersome to work with if you need to recover a large quantity of files. Like bootable backups, these are typically stored on external hard drives that need to be replaced on a regular basis.

In addition to the aforementioned Apple TimeMachine, the Windows OS also comes with a built-in service for creating incremental backups. Dropbox is a good option if you have a small number of files, but you may find the need to upgrade to a paid account if you have a large amount of data.

4.2.3 Cloud Backups

Cloud backup services like Backblaze or Crashplan offer comprehensive backup solutions for customers. These plans typically require a monthly subscription fee to maintain access to your backups. While bootable backups protect against hard drive failure and incremental backups protect against data corruption, cloud backups protect against catastrophic events like robberies, fires, and other natural disasters. A fire or a tornado that affect your house may destroy your laptop and any external hard drives you use for backup, but your cloud backup will be unaffected.

4.2.4 A Workflow for Backups

Just as we need a workflow for approaching file management, it is also important to establish a routine for backups. With backups, the most successful workflows are those that require next to no effort on your part.

If you primarily use a desktop, this can be as simple as leaving two external hard drives plugged into your computer since most backup software can be set to run automatically. If you have tasks that require you to manually do something (plug an external hard drive into your computer, for instance), create a reminder for yourself on a paper calendar or a digital calendar or to-do list application.

For this course in particular, it is *imperative* that you backup the data on your flash drive. A number of possibilities exist for accomplishing this:

- Keep a local copy of your flash drive's files on your computer.
- Keep a `.zip` archive of your files in a service like Dropbox or Google Drive. (Using a `.zip` archive will prevent issues with your `.git` repositories.)
- Maintain a second flash drive copies of all of your files.

Whatever solution you select, make sure you regularly update your backup. The more often you keep your backup archive updated, the less stressful and disruptive losing your drive will be. This will likely be a manual task, so follow the guidance above about creating a repeating calendar event or to-do list task reminder.

Chapter 5

Introduction to GitHub

Much of our interaction this semester outside of class will utilize GitHub.com (or just "GitHub"). GitHub is a web service that is a social network for programmers, developers, data scientists, researchers, and academics. It is also a tool for collaborating on projects, especially projects that involve writing code.

5.1 Git

GitHub is a web application that utilizes Git:

> Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Essentially, Git is a project-wide system for tracking changes to files. Think of it as Microsoft office's track changes feature on steroids - every change to every file in a directory (a "repository" or "repo" in Git-lingo) is tracked. You do no need to host files online to use Git. If you have a project saved locally (say, a doctoral thesis), you could utilize Git to version control that project without ever uploading it to the Internet.

For our purposes, this is just about all you need to know about Git. If you want to learn more, Git's 'About' page is a great place to start.

5.2 More Git-lingo

Beyond "repositories", there are a few additional terms that are specific to Git and that are helpful to know:

- **Clone**: Make an identical copy of a repository on your local hard drive.
- **Commit**: Approve any changes you have made to a repository.
- **Sync**: For cloned repositories, files that have been changed need to be synced or **pushed** to GitHub.com after they are committed.

5.3 GitHub.com

GitHub is a web service that can host projects using Git's version tracking. It is widely used by programmers, software developers, data scientists, and academics to host and collaborate projects.

GitHub is an excellent way to backup files for a project since you can "sync" changes made to a repository up to GitHub's servers. It is also an excellent way to collaborate on files with colleagues while also using

Git's version tracking. Repositories can be either public (like all of the repos for our seminar) or private, which means that only people who have been given access to can view the contents of the repo. Private repos require an upgraded account, which retails for $7/month.

Students can get access to GitHub's paid services for free, however, by signing up for a free student account. This will give you access to private repositories for as long as you are a student.

5.4 GitHub Repositories

Users of GitHub.com adhere to a couple of norms with their repositories that are worth knowing about. Repositories cannot have spaces in their names (much like variables in Stata), so the naming conventions that we will discuss in relation to Stata this semester all apply to GitHub as well!

Public GitHub repositories also contain (typically) at least three core files:

1. A **license** file - since the data is out there for public consumption, it is important to think about how that data is licensed. The norm among GitHub users has been to use open source licenses, which let others edit and adapt your work. There are a range of licenses that are commonly used on GitHub.

2. A **README** file - this describes the purpose and content of the project.

3. A **.gitignore** file - this stops certain types of files from being swept up by GitHub when a user syncs their files with a server.

Another norm is to write using a markup language known as Markdown. Markup languages allow users to specify exactly how they want their text to appear when it is parsed and processed by special software. This is different from, say, Microsoft Word, which is known as a "what you see is what you get" or **WYSIWYG** editor, which uses a graphical interface for constructing documents.

5.5 Storing GitHub Repositories

When you clone your repositories, you will be prompted to save them on your computer. There are a number of ways in which this process can introduce sources for trouble down the road:

2. Cloud storage services (Dropbox, Google Drive, etc.) - like external drives, these services can be a part of a backup workflow. However, like external drives, I have seen issues whee this has to contribute to sync errors with GitHub Desktop.

In order to avoid any issues, I suggest storing GitHub repositories on your computer's hard drive and not a thumb drive or other external device. Make sure your are saving your files in a place not backed up to Dropbox or another cloud storage service.

5.6 GitHub Issues

GitHub has a powerful tool for interaction called Issues. These can be accessed by opening a repository and then clicking on the "Issues" tab. Issues can be "opened" by anyone with access to the repository. They allow for a conversation to occur in the form of messages posted within the Issue itself. Files can be attached to Issues, and the messages can contain Markdown formatting. Once the conversation is complete, issues can be marked as "closed", which moves them into a secondary view on the website so that they are archived.

5.7 GitHub Desktop Application

GitHub Desktop is a tool that allows you to easily clone repositories hosted on GitHub, commit changes to them, and then sync those changes up to the website. You can also create new repositories, however this is not task you will have to do this semester. GitHub Desktop is not a fully functional desktop version of GitHub. For our purposes, it is important to note that the Desktop application will not let you easily identify when repositories have been updated by other users, view Wikis associated with repositories, or view Issues.

5.8 Learning More

GitHub has a resources page with links to websites that are great for helping you learn more about how Git and GitHub work!

Chapter 6

Final Words

We have finished a nice book.

Bibliography