Sociospatial Data Science

Christopher Prener, Ph.D.

2017-09-22

Contents

Preface **5**
License 6

1 Introduction **7**
1.1 What is data science? 7
1.2 How is this book organized? 8

I First Steps **11**

2 Getting Started **13**
2.1 Account Signups 13
2.2 Get Started with Software 14
2.3 Get Access to Books and Readings 14
2.4 Administrative Tasks 15

3 Approaching this Course **17**
3.1 Zen and the Art of Data Analysis 17
3.2 An Apple a Day 17
3.3 Reading with Purpose 18
3.4 Active Lectures and Labs 18

4 Protecting Your Work **19**
4.1 Data Management 19
4.2 Creating a Sustainable File System 20
4.3 Backing Up Your Data 23

Preface

This text is a companion text for both of my research methods courses at Saint Louis University:

- SOC 4650/5650 - Introduction to Geographic Information Science
- SOC 4930/5050 - Quantitative Analysis: Applied Inferential Statistics

The goal of the text is to create a reference for the intangible, subtle or disparate skills and ideas that contribute to being a successful *computational* social scientist. In writing this text, 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 this text 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.

In both classes, the course names are deceptive. We are not only concerned with statistical work or mapping. Rather, we are more fundamentally concerned with 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

[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 literate programming, which forms the basis of some of the practices we follow with `R`.

that we will discuss this semester are not taught as often in graduate programs, let along undergraduate 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

Chapter 1

Introduction

The first part of this text is designed to help get you oriented to coursework in computational social science. Both Introduction to Geographic Information Science (SOC 4650/5650) and Quantitative Analysis: Applied Inferential Statistics (SOC 4930/5050) are focused on building students' capacities to address social science research questions using tools that have been the traditional domain of computer and information scientists. The growth and application of these tools in a variety of disciplines both inside and outside of the social sciences has come to be known as data science. Both courses are taught from this perspective, so that while we focus on social science data, the tools and techniques are broadly applicable across disciplines.

1.1 What is data science?

Given data science's new emergence, its definition remains both contested and often unclear. For me, there are four key aspects to focus on when considering what constitutes data science:

1. Statistics
2. Programming
3. Visualization and Communication
4. Substantive Knowledge

I think of this as a "full stack" approach to computational research. You want to be well versed not only the techniques for generating and analyzing data, but also substantively in the academic literature about your area of interest as well as ways to communicate your findings with other researchers and the wider public.

1.1.1 Statistics

Statistics covers the mathematical techniques that we use to draw inference from our data. It is the main subject of one my courses, Quantitative Analysis. In Introduction to GIS, we do not explicitly cover much in the way of inferential statistics. However, the course is desinged to prepare you for a next level, Intermediate GIS course that covers spatial statistics.

1.1.2 Programming

Computer programming is an essential part of data science broadly and computational social science more specifically. Using a programming language means that our work can be easily reproduced. This emphasis on **reproducibility** is a response to a growing fear in many disciplines that results are replicable from study

to study, which raises questions about the validity of much of the research work that we do. Our goal in both courses is to produce research that is as reproducible as possible.

This book introduces two programming languages - `R` and Python. In Quantitative Analysis, we will be focused exclusively on learning `R` and using it to produce statistical analyses. In Introduction to GIS, we will spend a lot of time in `R`, but we will also use some Python to help pass data from `R` to ArcGIS, the mapping application we will use. I will also provide some additional Python lessons to folks who are interested, but these will not be required for the course.

1.1.3 Visualization & Communication

Visualization is a fundamental aspect of data science work. It is how we make our results easily digestible and accessible to a wider audience, many of whom may not be able to interpret statistical output but can learn from a well-designed scatter plot. In Quantitative Analysis, we will focus on building plots to communicate information about statistical distributions and the relationships between our variables. In Introduction to GIS, we will use the same fundamental skills to build simple maps in `R`. We will extend our emphasis on visualization to ArcGIS, where we will focus on producing cartographically rich depictions of our data.

Separetely, we will discuss the presentation of statistical data in tables using LaTex in Quantitative Analysis as well as the producing to conference-style presentations to communicate research findings. In Introduction to GIS, we will focus on producing conference-style posters instead. These are different mediums, but they rely on the same design fundamentals that are covered in this text.

1.1.4 Substantive Knowledge

Substantive knowledge covers two tangentially related topics: the ability to work well in groups and the ability to digest and integrate an academic literature into your own research. Each of these topics receive some focus in both Quantitative Analysis and Introduction to GIS. Each class has some group work associated with the completion with weekly lab assignments. Introduction to GIS also has a group work component associated with the final project. In each class, students' final projects are focused on a specific content area that requires at least some background research and knowledge. Synthesizing this knowledge and integrating it into your final projects is a key piece of addressing this facet of data science.

1.2 How is this book organized?

Like many books about data science, this text follows a number of standard conventions related to how it is organized and how examples are given. This section introduces those conventions.

1.2.1 Typefaces and Fonts

Technical publications that describe scientific computing processes use a `monospaced typewriter style typeface` to refer to commands (inputs) and results (outputs). In some documents, like lecture slides and cheat-sheets, I may highlight a command by using a particular color to increase the visibility of the command name itself.

The `typewriter typeface` is also used to refer to functions (e.g. `library()`), filenames (e.g. `mpg.csv`) 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).

Technical 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,

`str(dataFrame)` (with `dataFrame` *italicized*) indicates that you should replace the text `dataFrame` with the appropriate variable name from your dataset.

These publications also use a sans serif typeface to refer to areas of the user interface, menu items, and buttons. I cannot replicate that here because of the publishing software that I use, but you'll notice this text in course documents. We will therefore use the `typewriter typeface` in the User Guide to identify these same features.

Technical documents also use a sans serif or `typewriter` 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`).

1.2.2 Data

There are two sets of data that are used in this text, and they are also used as part of Quantitative Analysis and Introduction to GIS. Both are available as `R` packages. The `testDriveR` package contains some generic data that are particularly well suited for exploring statistical topics. The `stlData` package contains data that can be mapped at various levels. Both of these packages are currently available on GitHub, and can be installed using the `devtools` package:

```
install.packages("devtools")
library(devtools)
devtools::install_github("chris-prener/testDriveR")
devtools::install_github("chris-prener/stlData")
```

1.2.3 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`:

```
> library(stlData)
> str(stlLead)
'data.frame':   106 obs. of  15 variables:
 $ geoID         : num  2.95e+10 2.95e+10 2.95e+10 2.95e+10 2.95e+10 ...
 $ tractCE       : int  118100 117400 126700 119102 126800 126900 108100 127000 127400 103700 ...
 $ nameLSAD      : chr  "Census Tract 1181" "Census Tract 1174" "Census Tract 1267" "Census Tract 1191.
 $ countTested   : int  345 871 458 182 486 1296 903 585 2116 417 ...
 $ pctElevated   : num  9.57 12.06 18.12 2.2 4.73 ...
 $ totalPop      : int  1161 4307 1089 3237 3490 4590 3144 2052 5486 2408 ...
 $ totalPop_MOE  : int  192 447 199 309 231 826 464 273 516 274 ...
 $ white         : int  414 2604 432 2008 3026 148 108 304 1777 2149 ...
 $ white_MOE     : int  100 303 116 262 270 217 111 82 391 212 ...
 $ black         : int  724 1338 631 646 194 4320 3020 1739 3603 156 ...
 $ black_MOE     : int  179 374 187 210 98 760 442 283 621 190 ...
 $ povertyTot    : int  324 615 506 958 349 1743 652 331 2524 254 ...
 $ povertyTot_MOE: int  140 255 164 234 129 825 305 156 598 88 ...
 $ povertyU18    : int  109 169 98 15 35 627 256 47 1110 15 ...
 $ povertyU18_MOE: int  105 156 60 25 47 595 136 79 318 23 ...
```

Examples will almost always use the dataframe `stlLead`, which comes with the `stlData` package. To open it, simply load the `stlData` package using the `library()` function and then start referencing `stlLead` anytime you need a dataframe. This allows you to easily recreate examples by minimizing dependencies within your code.

Part I

First Steps

Chapter 2

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!

2.1 Account Signups

2.1.1 Get Started with Slack

We'll be using the messaging platform Slack as a space for "virtual office hours". Slack is a messaging system used by teams of all kinds. If you can text, you can use Slack. You will need to sign-up for the SOC 5050 Slack organization here. You will need to complete the signup process even if you use Slack for other purposes. Consider installing either the desktop or the mobile apps for Slack to keep in touch and receive push alerts!

2.1.2 Get Started with GitHub

The website that is hosting this wiki is called GitHub. GitHub is used by programmers, data scientists, and researchers for hosting computer code, data, and project materials. We will be using GitHub extensively this semester. You will need a free account, which you can sign up for one from GitHub's homepage. If you already have a GitHub account, you do not need a new one. *Once you have a GitHub user name, send Chris a Direct Message via Slack with it so that you can be added to the SOC 5050 organization.*

2.1.3 Get Started with LaTeX

We'll be doing a little bit of writing using LaTeX, which is a markup language that makes technical writing easier. We'll be using ShareLaTeX this semester for this purpose. ShareLaTeX is a bit like Google Docs, but for LaTeX. It is a "freemium" service - please don't pay for any additional features - you won't need them! You can sign-up for ShareLaTeX on their website.

2.2 Get Started with Software

If you will be using your own computer in class, you'll want to install a number of applications. If you aren't using your own computer, you can skip this section! All of these applications are available in our classroom, and - lucky you - you get 24-hour access to Morrissey Hall for the semester.

2.2.1 Computer Prep

Before you install your software, you should do the following:

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. You can get anti-virus software for free from SLU. Go to `ITS Software Downloads` under `Tools` on mySLU.
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.

2.2.2 Software Installation

Now that your computer is up-to-date

1. The computing language `R` needs to be downloaded and installed. You can download it from the University of California-Berkeley. Choose "Download R for (Mac) OS X" or "Download R for Windows".
2. RStudio is a graphical user interface for `R` that will make learning the language and using it much, much easier. You should download the *free* version of RStudio from their website. Choose the installer for your platform, and ping me on Slack if you have any questions.
3. GitHub Desktop is a client for interacting with GitHub that makes downloading and uploading files a breeze. You can download it from the developer's website.
4. Atom is a text editor that is produced by the same folks who operate GitHub. Download Atom from the developer's website.

2.3 Get Access to Books and Readings

2.3.1 Books

There are three books required for this course. Each book has been selected to correspond with one or more of the course objectives. The books are:

1. Freedman, David, Robert Pisani, and Roger Purves. 2007. *Statistics*. 4th edition. New York, NY: W.W. Norton and Company.
2. Wheelan, Charles. 2014. *Naked Statistics: Stripping the Dread from the Data*. New York, NY: W.W. Norton and Company.

3. Wickham, Hadley and Garrett Grolemund. 2016. *R for data science*. Sebastopol, CA: O'Reilly. Webbook Available.

All of the books are available in the bookstore. They can also be ordered online. If you would rather use ebooks, those are acceptable for this course as well.

2.3.2 Check Out the Readings for Week 01

All but one of the Week 01 readings are available on our course's electronic reserves site, and the password is posted in Slack on the `#helpdesk-coursework` channel. The initial section of Wickham and Grolemund can be found via the webbook.

2.4 Administrative Tasks

There are two forms that all students must fill out by Tuesday, September 5th:

1. the Student Information Sheet, which gives me some info about you and gives you the chance to let me know about any initial concerns you might have.
2. the un-graded Diagnostic Assessment, which is designed to get a sense of where each student's math skills are currently. Please don't consult outside materials as you do this - if you are not sure how to answer, make the most educated guess you can and move on. If you look answers up it defeats the purpose of this exercise!

Chapter 3

Approaching this Course

Students have varying experiences learning statistical techniques. For some, the math and programming that are the foundation for modern statistical 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.

3.1 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. `R` synatx, LaTeX commands, and Markdown can seem like foreign alphabets at first. Debugging `R` synatx 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 statistics coursework is often a much better approach than trying to power through problems.

3.2 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 chunks - 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:

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

- Friday: First reading
- Saturday: Second reading

3.3 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.

3.4 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 `R`, you can look this information up. Conversely, detailed notes on executing `R` 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.

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 Data Management

One of the themes in "Good Enough Practices in Scientific Computing", referenced in the previous chapter, 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 expansive - 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. For that reason, we'll give you not just the course data but a read me file and a metadictionary that lists all of the files we've disseminated to you.

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 opened 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 desirable for storing data.

Both tabular and spatial data, in their final forms, should be what we consider "tidy data"[1] Tidy data are defined by a number of common attributes - each column represents a single variable or attribute and each row represents a single, unique observation. This arrangement should produce clear, easy to read datasets that represent a single observational unit.

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`).

4.2 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.

4.2.1 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`!).

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

4.2.2 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
  /Software
  /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 (`FinalProject` or `ProblemSets`), 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, try to 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`.

4.2.3 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-03**, when it is **cloned** from GitHub. A cloned directory is one that retains a digital 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.**

4.2.4 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).

4.2.5 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-03**. 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 in your folder structure.

4.2.6 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.

4.2.7 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.

4.2.8 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.

4.2.9 The `Notes` Directory

Use this as a home for course notes.

4.2.10 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.

4.2.11 The `Readings` Directory

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

4.2.12 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.

4.3 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.

4.3.1 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.

4.3.2 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.

4.3.3 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.

4.3.4 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.

Bibliography