

Technical Data Skills for Reproducible Research

Harrison Dekker and Paula Lackie

The fact that raw data is rarely usable for analysis without significant work is a point I try hard to make with my students. I told them “do not underestimate the difficulty of data preparation.” When they turned in their projects, many of them reported that they had underestimated the difficulty of data preparation.

~ David Mimno, Assistant Professor, Computer and Information Science, Cornell University.¹

STATISTICIANS NICHOLAS HORTON, Benjamin Baumer, and Hadley Wickham state that “statistical educators play a key role in helping to prepare the next generation of statisticians and data scientists,” but “there are barriers and time costs to the introduction of reproducible analysis tools and more sophisticated data-management and -manipulation skills to our courses.”² The time-consuming nature of data preparation, a precursor to reproducible research, is not just an issue in the introductory curriculum. Professional data scientists also report “spending most of their time ... turning data into a usable form rather than looking for insights.”³ Given their research experience and the allocation of precious time and attention in courses, it is not surprising that faculty members are wary of losing focus on the richness of the analytic process by diving too deeply into the time-consuming and distracting processes of data preparation. However, this avoidance undervalues the benefits of understanding the unique characteristics of the data before jumping into the analysis stage. As a result, novice researchers rarely have a chance to systematically learn about the essentials of data preparation before they are faced with applying the necessary principles in their own research.

While one traditional role for data librarians is to assist researchers by collecting, locating, and recommending appropriate data resources, the data service provider (often a data librarian with additional technical data experience) can offer equally important services to help researchers bridge the gap between acquiring and analyzing their data. These data professionals must field a wide range of research questions and a wider range of data analysis and formatting issues with the ultimate goal of reinforcing reproducible research practices. Filling this niche can be especially rewarding, largely due to the high degree of creative problem solving that it requires.

A fundamental characteristic of the technical data preparation component of data services is that unlike statistical methodology, which lends itself to traditional instructional methods (i.e., in which foundational skills are taught and new ones progressively layered on, thus gradually building the researcher's set of tools), the ability to "wrangle"[†] well develops more organically. As in the maxim "that which doesn't kill you, makes you stronger," data preparation skills are strengthened as the wrangler grapples with a wider range of "messy" data problems.

As data services providers, we frequently observe one common source of messy data: idiosyncratic data management practices. With experience we can anticipate when and how these unique data preparation processes will become problematic for researchers and alert them to these probabilities. Since "reproducibility is collaboration with people you don't know, including yourself next week,"⁴ getting researchers started early with best practices facilitates the diffusion of reproducible research practices.

To develop best practices, novices generally need to learn from experts. In this case, because the norms of record keeping (i.e., documentation) are often not taught in the curriculum, data services professionals with expert knowledge will find themselves in that sweet spot in what Everett Rogers calls "the diffusion of innovations," the place in the cycle also known as the "take off point" where change agents have the greatest impact on the adoption rate of an innovation.⁵

Data services professionals can increase the adoption rate for conducting reproducible research by modeling best practices and filling the data preparation instruction gap.

So what are these best practices? And how can they be classified? In short, they represent a systematic approach to managing and documenting all components of the research process such that a project may be set down by one researcher and picked up by another, who will not only be able to make sense of it but also be

[†] With the dramatic pace of development in data science, the vocabulary surrounding it has not yet settled and several terms relate to overlapping aspects of the work. Data wrangling here refers to the gathering and managing of disparate data sources and file types, but it can also include wrangling individual variables. Data science is the whole set of skills necessary to extract knowledge from data, usually from disparate sources.

able to completely replicate the scholarship as presented.[‡] These principles need to be applied to all stages throughout the entire lifecycle of that research. To itemize best practices, we can look at an ideal, generic research project and separate it into stages. At each of these stages and for all layers of the work there needs to be the following basic best practices components: 1) metadata and other documentation; 2) file naming and format consistency; and 3) a deliberate strategy for organizing and backing up files.

The Basics of Best Practices

We begin with **metadata and other documentation**: the plain language description of where all data resources and specific variables came from and what they mean. It is important that this information be gathered as, or if possible before, the data are collected. This includes the proper, full citations for each data resource (especially in secondary research) as well as any related details about the variables. Although this step is basic, novice researchers eager to get into the analysis stage often overlook it as superfluous. It is not until they need to describe their datasets and provide adequate citations in their research papers that novices realize how difficult, time consuming, and sometimes impossible it is to compile this information after the fact. In practice, this process is similar to managing standard footnotes and bibliographies in traditional research as references are collected and used, but novices do not know to apply these techniques to working with data. For the data support professional, advising on these forms of documentation in a timely fashion is the easiest and most basic step in supporting researchers working with data.

Additional documentation can be in `readme`[§] files, comments and annotations in code files, and ideally, in an additional project research journal. A research journal is a simple record of potential hypotheses with initial justifications, thoughts about primary or secondary resources, and a place to track search strategies. The journal helps a researcher to discover the aspects of her research that she does not yet understand. It is especially useful for novice researchers to maintain a research journal in which they track their progress in language that is meaningful to them. For faculty working with research assistants, reviewing the research journal can lead to cleaner and more rapid uptake by assistants new to the project. As novices mature through their research project and practice as researchers in general, they

‡ Because of the increase in service demands on faculty is often correlated with a degradation in their research time, the person most likely to pick up where they left off is themselves, sometimes after years of hiatus. Changes in demands on faculty should be a powerful motivator for adopting best practices in reproducible research over less-documented idiosyncratic research processes.

§ By convention, a `readme` file is a small documentation file describing the contents of other files in the same directory or project and assigned a name like `README.txt`, `READ.ME`, `README.1ST`, etc.

will develop the expertise to recognize which aspects are most useful to record and in the associated expert vocabulary.

Next is clarity and consistency in **file naming conventions and formats**.⁶ Naming a file *Final* is rarely accurate or useful; a better choice would be *TITLE_submitted_to_X_on_DATE.pdf*.[†] The same is true for writing code; *Data.sps* is not as useful as *Merge_code_for_NES2012to14_12June2015.sps*. Meaningful file names are another form of metadata useful to the original researcher or others attempting to make sense of a project at some later point in time, particularly when files have been moved out of the context in which they were created.

There also needs to be clarity and consistency in **file organization** (e.g., box.com or a local drive for working copies), and only one location (with verified backups) for the authoritative version. We are particularly fond of Git and GitHub for documentation, code, and sometimes even the papers.[‡] Being diligent about following through with documented file locations is especially important when working in groups.

The last component is a **backup strategy**, to recover from immediate file corruption with minimal loss of work time, as well as file snapshots, to allow for the ability to backtrack in a process and recover some specific content that had been removed. Ideally this strategy will include a definition of when these materials may be safely discarded as well. Note that some services provide a file recovery option, but these are not infallible. For instance, with a free Dropbox account, if an error occurs but is not noticed within the 90-day recovery window (e.g., from one term break to the next) it is not possible to use that avenue for file recovery. In another example, with Google Drive it is important that the Principal Investigator retain full ownership and not just editing ability for all related files. Research assistants come and go, and if a Google account is deleted, there is no way to recover any Google Drive files that person *owned*.

Advanced Best Practices: Working with Code Files

The premise behind these best practices for file management and documentation apply when working within code files as well. We recommend that novices especially comment (i.e., annotate) *in plain language* within the code file what the code does, what inputs are required, when the code was written, and by whom the code was written. Explaining what it is they are writing in code will facilitate their own reuse as well as contribute to the process of learning to code and to conduct repro-

[†] For milestone versions of the research and outcomes the files should be in at least two formats: PDF and the format of the editable version.

[‡] Note that best practice for the data file/s involve one additional step, storing a read-only copy of original data files before any cleaning or transformation steps have taken place.

ducible research. In addition to commenting, it is also good practice for users to become aware of the style guides for each code language.⁷

Good style is important because while your code only has one author, it will usually have multiple readers, and when you know you will be working with multiple people on the same code, it's a good idea to agree on a common style up-front.

~ Hadley Wickham, Chief Scientist at RStudio, Inc.⁸

These sorts of practices are a few of what some, like author Scott Long, call *data workflow practices*.⁹ The ability to develop and apply good workflow is contingent on a variety of factors. For one, researchers must follow a set of best practices like those outlined earlier, which allow them to record as much of their data manipulation and analysis processes as possible. Writing well-annotated code files (as opposed to utilizing only a click-based interface) is the most efficient way to document the research process. For another, it is critical that researchers see these skills within the larger context of an overall ethos of reproducibility. Data service professionals can reinforce the ethos by reminding researchers that will benefit from reproducible practices. The immediate benefits will be measurable in terms of improvements to their productivity, for instance by making it possible to reconstruct the logic of a past data workflow. Additionally, researchers can be reminded that a possible long term benefit of reproducible practices is an improvement to their academic reputation because these practices facilitate replication of their work by others. Working with reproducibility in mind addresses directly the replication crisis in social science research.

Mitigating the Replication Crisis through Use of Best Practices

There is a replication crisis throughout research. When subjected to independent scrutiny, results routinely fail to stand up. We are starting to accept that there will always be glitches and flaws. Slowly, as a consequence, the culture of science is shifting beneath everyone's feet to recognize this reality, work with it, and create structural changes or funding models to improve it.

~ Ben Goldacre, Senior Clinical Research Fellow, University of Oxford.¹⁰

While there is alarm over the “replication crisis” in research, this has encouraged pragmatists into action. Some of this has been facilitated in the United States by the addition of required data management planning or data sharing for certain federal grants and academic publishers as well as widespread training activities initiated by leading research data archives and professional associations worldwide.[†] On the curricular level, efforts like the Teaching Integrity in Empirical Research (TIER) project at Haverford College freely provide their curricular protocols for a thorough approach in training for “complete replicability.”¹¹ As is noted on the TIER website:

The guiding principle is that the information included in the documentation of a statistical study should be complete and transparent enough to allow an interested third party to easily and exactly reproduce all the steps of data management and analysis that led from the original data files to the results reported in the paper. By contrast, among most academic journals that maintain on-line repositories of documentation for the empirical studies they publish, the requirement is just that authors submit the processed data (after cleaning, merging, etc.) used for analysis, and computer code with commands that generate the results in the paper from the processed data. This standard of “partial replicability” leaves the steps needed to transform the original data files into the files used for analysis entirely undocumented.¹²

The TIER research framework is broadly useful both as a guide for students as well as a framework for developing curricular support materials by faculty or data support professionals. The main elements of the documentation specified by the TIER protocol include:

- **Keep a read-only copy of all the original data** files in from which any data used in the study were extracted. In some cases, it is necessary also to save metadata with coding or other information explaining how to interpret the data in the original files.
- **Maintain metadata files** containing additional information a user would need to understand the data, such as variable definitions, units of measurement, coding schemes, and sampling methods.
- **Provide all commented command files** (i.e., code) necessary to construct the final dataset(s) used for analysis, and finally to generate all the statistical results—the numbers reported in text, tables and figures—

† Join the international organization *IASSIST Data* to tap into this robust and growing community. (<http://iassistdata.org>).

presented in the paper. These files will include all the original instructions needed to both access the data in the original files and process it as necessary (cleaning, merging, defining new variables, etc.). Also include a readme file that gives instructions for using the data and command files to replicate the results reported in the paper.[‡]

While basic, these steps are rarely practiced. There are rational reasons for this. The researcher may be unclear on the utility of documentation or code for her own use or underestimate the amount of trial and error effort necessary to complete the data preparation process. These tendencies to skip documentation steps in the service of expediency provide another opportunity for timely intervention by a data service professional.

It is common for novices not to realize how frequently data preparation steps must be re-done due to data changes or discovered problems. In other cases, they are new to the conventions of coding and do not realize the ease with which commands can be saved and reused. This problem is compounded when in their preparatory courses they choose use a spreadsheet package for analysis or a statistics package that steers them towards menu driven operations. Reproducibility is hindered by the immediacy of less technical “simplified” but opaque processes.

Technical Components of Data Preparation

To provide basic technical assistance, a data services professional should be familiar enough with the common components of technical documentation to interpret and explain to a researcher. For novice researchers or more experienced ones who are expanding their research to unfamiliar domains, the expertise of a librarian who has experience with a broad range of documentation can be critical in determining whether they will be able to use a particular dataset. On a more technical level, this documentation is often essential for understanding what steps are needed to process the data in a statistical package.

In the social sciences, an example of technical documentation that almost always accompanies a published dataset is the statistical codebook. The lack of stan-

‡ Corporate software trademark rules as well as open source logic generally reinforce the best practice of including notation that indicates precisely which version of which software, with which add-ons, produced the results (e.g. “the suggested citation for the Stata 14 software is StataCorp. 2015. Stata Statistical Software: Release 14. College Station, TX: StataCorp LP.” <http://www.stata.com/support/faqs/resources/citing-software-documentation-faqs/>). Some academic communities include code for citing the specific versions and packages used in the analysis. See, for example: <http://astrostatistics.psu.edu/su07/R/html/utlils/html/citation.html>. Details about the software used are integral to complete replicability.

dardization in layout and content can make codebook interpretation a challenge for both novice and experienced researchers. Still, a codebook can generally be relied upon to provide standalone documentation of such technical details as the layout, structure and content of a dataset. In terms of subject matter, the codebook can provide critical information such as whether it is a sample or population; a list of variable names and labels; the values assigned to categorical variables and the corresponding value labels (e.g. 1 = “Male,” 2 = ”Female”); and how missing values are represented. All of these are important when using the data in a statistical software package. Codebooks may also contain other types of information of importance to researchers. For instance, in survey research, the codebook will usually contain the original text of the survey questions as well as details on how the survey was conducted and the sampling methodology employed. A codebook often includes a data dictionary, a kind of census of the basic elements of the data file (or files in a database). This includes the basic characteristics of each variable, the relationships to other variables (often called “fields” in programming), and their format.

The following example taken from the 1993 *Indonesian Family Life Survey*¹³ demonstrates some of the challenges inherent in codebook interpretation (see Figure 7.1). The excerpt at first glance may seem cryptic, but it contains critical details about the file layout. Without this information (or an accompanying complete script file in an accessible code language), the original data are useless.

Figure 7.1. Codebook Example

Variable Name	Location	Length	Decimals	Format	Class	Variable Value	Label
AR22	188	2	0	N	S	PRIMARY ACTVTY values:	DURING PAST WEEK label 1 WORKING/EARNING INCO 2 JOB SEARCHING 3 ATTENDING SCHOOL 4 HOUSEKEEPING 5 RETIRED 6 OTHER, SPECIFY
						ranges: lower	upper
						1	6
						96	99

- In the first column is the Variable Name (AR22), which serves as a unique identifier for this set of information in the dataset. The variable name is the key to finding and using these data after the dataset has been successfully loaded into a statistics package and for locating other relevant metadata in the codebook or other documentation sources.[†]

[†] Note that clever novices will soon learn to modify the variable names to suit their work, but it is wise practice to retain the originally assigned variable name so that it can be tracked throughout the codebook and other documentation. It is also helpful for reproducibility.

- The second, third, and fourth columns (Location, Length, and Decimals), provide important metadata about the actual layout of this variable in the original data file. In this case, the original data file is in a *fixed width* format. The location (188 characters from the left), length (2 characters wide) and decimal field (no decimal places) are essential for writing code to import (or to read) the variable into a statistical package. This is because in a fixed width file, there is nothing in the file to delimit (separate) each variable from the next within the original data, which may include numbers or text.
- The fifth column (Format) tells us that the variable is intended to be numeric (N), so if we find any non-numeric character in this variable, there is a problem with the file that needs to be sorted out before any useful analysis may be started.
- The sixth column (CLASS) is a variable that only has relevance to the program used to prepare the dataset and is irrelevant to most analysis, a detail documented elsewhere in the codebook.
- The seventh and eighth columns (Variable Label and Value Label) do not follow the pattern set in the previous columns and instead stack the information horizontally: Variable Label = “PRIMARY ACTIVITY DURING PAST WEEK” and value label = the plain text explanation for what each assigned value means (e.g. 1 = WORKING/EARNING INCO). It may be helpful to note that formatting issues like this are not unusual.

Although all of the information has utility, the variable label is the portion in which the researcher is most interested. It provides a longer description of the variable than the Variable Name, which in this example actually provides no descriptive information. Most statistical software packages allow the variable label and the equally important value labels to be stored and associated with a specific variable name when the dataset is saved in the native format of that package. Importantly, without a dataset saved in native format and without a script for assigning this information to the data set, the codebook is the only source for assigning meaning to the numbers.[‡]

Providing assistance with codebook interpretation is an experience that all data librarians are likely to encounter given the popularity of data archives like the Inter-university Consortium for Political and Social Research (ICPSR) and the UK Data Archive, which bundle codebooks with virtually all their published datasets.¹⁴ One bad habit that a service provider should be mindful of is the tendency of researchers to completely bypass the codebook, like most of us bypass software

[‡] Often there is an associated “data dictionary” as well. The standards as to what each of these metadata resources provide are inconsistently followed and/or differ among communities of practice.

manuals, and expect to “open the file” and be able to ascertain by “looking at the data” whether it will be suitable for their analysis. A timely intervention in this scenario is another opportunity to teach basic documentation reading skills and simultaneously assess the researcher’s level of expertise and future need for assistance.

Another common and potentially more challenging interaction for data professionals comes when a researcher requires assistance with data acquired from a less curated source where the documentation is not as neatly bundled with the data. Examples of these are acquisition of under-documented data from another researcher or from online databases such as *Social Explorer*, *World Development Indicators* or *Datastream*. These latter systems allow a researcher to extract particular variables and other subsets from a large database system through a user-friendly interface. To a certain extent, these systems eliminate the need for some types of technical documentation, such as file structure characteristics. The extracted data are typically saved in a spreadsheet or comma-delimited format and thus easy to open and inspect in a familiar software package like Microsoft Excel. But it is not uncommon to encounter situations requiring more documentation in order to, for instance, answer questions about the source of the data or interpret certain variables. This type of “metadata sleuthing” will be familiar territory for an experienced librarian.

It is tempting to think that once a researcher has her data “in hand” and has sufficient documentation to enable its use, the data professional’s job is done. In practice, the opposite is often the case; the creative work of data preparation has just begun and so has the potential need for assistance. Following are descriptions of useful technical skill areas for supporting researchers working with data. When providing this type of assistance, it is important that data professionals resist the urge to prepare the data for the researchers and consider the long-term value of educating researchers about reproducible practices by assisting them in doing this work themselves.

To this end, particular emphasis should be placed on writing code. Researchers should be encouraged to learn to use commands, rather than relying solely on graphical user interfaces, to perform data preparation and analysis. All the major statistical software packages, namely SAS, SPSS, Stata, and R, allow or require the use of commands. To promote reproducibility, all commands used to accomplish a task or a series of tasks should be saved in a file, (or *script*), which can then be processed by the statistical software package. Each package has its own command language, and while becoming proficient enough to provide assistance in multiple languages may seem a daunting task at first, by focusing on a narrow set of the most commonly used skills—like the ones outlined below—and by identifying useful sources of help, it is a surprisingly achievable goal.

Conversion

It is common practice for data archives, such as ICPSR, to distribute data in multiple formats. The most basic is an archival format (like ASCII) provided with a script for loading (i.e., *opening*, or *reading*) the file into their preferred software. Data archives may also provide the data in any number of statistical package formats, making it easy for researchers to simply open the file in the package of their choosing. This, however, is the best-case scenario. In other cases the researcher's data may be provided with appropriate metadata, but the data file is in a format that is incompatible with the required analysis software. There are as many variations to this problem as there are file types, so it is not possible to provide a comprehensive set of solutions here.

Most conversion issues can be solved by using the Stat/Transfer software package. This extremely flexible and simple application should be available to every data services professional and is relatively inexpensive. There are no clear open source alternatives, although open source programming languages often used for data wrangling, like R, Python, and Perl, have the capability of opening and processing a wide range of file formats. This type of solution requires the user to know how to write code, which even with these skills will take significantly longer than Stat/Transfer.

While software tools are an important aspect for data conversion, another equally important aspect is knowledge of various file formats. Novice users often routinely open files by double clicking them. This approach typically works only when a file is saved in a proprietary format, the software in question is loaded on the machine, and file associations are set up correctly, which may not be the case for obscure data formats. Therefore, a fundamental concept to teach users is to open data files from the application menus, or better still, from commands or a command script written in the language used by the statistics package.

In summary, when a user asks for assistance with file conversion, the initial assumption should not be that they will need to use a separate application to do the job. Generally, the file extension identifies whether it belongs to a known proprietary software. If the extension is unfamiliar, an online resource like the Wikipedia "List of file formats" page (https://en.wikipedia.org/wiki/List_of_file_formats) should help identify the software with which it can be opened. If it is not an identified format, a next step is to try to open the file with a programming text editor like Sublime or Textmate, or to view the first few lines with a command line utility like "head" or "less" on a Mac or Linux machine or "more" on Windows. When the associated documentation fails it is often possible to discern the format of the data (assuming that it is not in an unreadable binary format) by looking for clues in the file itself. Another approach is to examine accompanying script files, to sort out any codebook-like information such as variable names and datatypes.

Importing

As indicated above, importing data is sometimes an alternative to converting data. Generally speaking, importing data into a statistical package means that the data are being read from a format other than the package's native format. Virtually all statistical software can import (*or read*) data from standard text formats like comma delimited or fixed width,[†] and many include the ability to import data saved in the native format of some other statistical packages.

The import process is often managed via a wizard, in packages using a graphical user interface, or via scripted commands. Scripted import commands often require the user to specify various command options. For instance, an import command usually has a way to specify that the first line of the data file contains variable names, or that the file should be read starting from a line other than the first one. Reading the documentation is the key to understanding the various options available and how to use them. Inspecting the contents of a file with a programming editor or command line tool before trying to import it is another best practice, especially when the file is assumed to be in ASCII format.

The code-approach to importing data is preferred for several reasons: it nudges a user towards a replicable workflow; it facilitates her own record keeping; and it helps her understand the workings of the research process. If a file is opened via code which is then saved to a research process directory, there is an exact record of which file was opened, where it was saved, and what command parameters were required to open the data file.

One last best practice with importing data worth mentioning is *patience*; depending on file size and other factors, the initial import of a file can take anywhere from several seconds to several hours. Data users will sometimes report that they attempted to load a large file and it “crashed” their computer. It is important to ascertain details like the amount of system memory (RAM) available, the size of the file, and whether the software they were using has any file size restrictions in order to rule out the possibility that they simply did not wait long enough.

Validation

Once the data file is in place it is time to look at the individual variables. Since it is common for data services professionals to provide assistance to novice users who are loading a dataset for the first time, this is an ideal time to remind them that there are important validation steps to consider before they plunge into analysis. Validation is

† A *comma-delimited* file refers to a file with the extension .csv (comma separated values—where each piece of information is separated from the next by a comma.) Fixed width data require a codebook or some basic programming to separate the raw numbers into separate variables (or fields). These data will be in the most basic ASCII format and the extension is likely to be .txt.

accomplished by running basic descriptive analyses on all of the relevant variables. At a minimum, variables should match what is reported in the documentation. Basically, the goal is to determine if the researcher actually has the data that they think they have.

While there are some essential components of variables that are common across academic disciplines, these components are not always identified by the same terms. Regardless of the vocabulary used, there are still some basic characteristics of variables that are fundamental to statistical analysis. This list is roughly in order of their level of importance.

1. Data types

The most fundamental distinction between data types is whether the variable is numeric or string (text). If numeric, a second step involves specifying the specific type of numeric variable. There are three basic numeric variable subtypes (continuous, ordinal, and categorical), but the names for these categories vary by disciplinary norms.

A note about numeric variable subtypes:

Continuous (i.e., interval or scale, and ratio, which has a definitive 0 point) variables are most easily characterized by the conceptual notion that the intervals between each possible measured point are consistent. (The distance between 1 and 2 centimeters is the same as the distance between 20 and 21 centimeters.) Interval variables have the highest explanatory power in statistics.

Ordinal variables are merely variables where the values are in a specific order. Think of a Likert scale (e.g. 1=never, 2=rarely, 3=occasionally, 3=often, 4=very frequently) where the value of 4 is more than the value of 1, but is it exactly 4 times more? Ordinal variables can provide some inference to the direction in a relationship between variables.

Categorical (or nominal) variables are values that might as well be randomly assigned. Think of a variable “color” where 1=blue, 2=purple, 3=brown, or a zip code variable. Computing the average is meaningless on a categorical variable. Categorical variables can only indicate whether a relationship exists between variables that is more or less likely than random chance.

Dummy Variables: a special case variable type where the only two values possible are 1 and 0. In this case, a categorical or ordinal variable may be converted to a kind of continuous variable by splitting the relevant values into separate variables where 1=*it is* and 0=*it isn't*.

Type assignment is one step in this process. It refers to setting the appropriate data type for each variable in a data set. The codebook is important to this process because it should list the proper variable type for each variable. The codebook is not infallible, however, as some data providers will simply supply an unedited software-produced data dictionary that includes inaccurately assigned data types, or for various reasons the data do not match what is defined in the codebook. Verification that a variable is what it is expected to be can be accomplished through a simple univariate analysis for each variable of interest.

Another important data type is date. When date variables are read from a file, in most cases they are interpreted as strings (e.g., “8/19/2015” or “19 August 2015”). Fortunately, a common feature of statistical software is the ability to convert strings containing consistently formatted dates into a date data type. Whenever possible, these conversions should be made to ensure that dates will sort properly and that the data will be compatible with other date-specific functionality. Note that date types are capable of containing a time component as well, but it is not required.

In some cases, particularly when data pass through Excel on their way to a different data analysis program, the data type can be incorrectly assigned and the data may even be modified from the original format. This can be difficult to identify without careful validation of all of the data types, since a single number among thousands could be auto-modified from a number to a string (text), and that will typically convert the entire variable type into a string format (even if it looks like it is all numbers) that is unsuitable for most statistical analysis. For instance, Table 7.1 shows Excel auto-convert examples:

Table 7.1. Examples of Auto-converted Data in Excel®		
Data as entered	Auto-converted by Excel®	Effect on data
44:45	44:45:00	Converts to timestamp with special formatting (hh:mm:ss)
8-3002	Aug-02	Converts into a “date” with text that replaces the original data

When unexpected character values are discovered in a variable, the researcher must determine whether this is an error (messy data) or whether the value has some significance. Again we refer to the codebook for clarity. If there is no codebook it is even more important to examine each variable individually to assure that all values within each variable fall within the expected range. Data outside the expected range may be errors, intended “missing” values, or outliers, which are valid data with values that could skew the analysis because they are dramatically outside the norm.

All of this activity may be classified as *data cleaning* and is best managed through writing code. While some data cleaning is possible within statistical packages, it is often too cumbersome to be practical. An alternate approach is using a program designed for this process (e.g., OpenRefine and Colectica's free Excel add on), though careful documentation is essential to ensure reproducibility. For more efficient data cleaning and transformation work, the Python programming language is recommended because it is a powerful and relatively simple programming language.

A note about missing values:

Missing values may be assigned by the software or programmed by the researcher. The key is in recognizing which variables may need to have missing values identified. An example is a missing value indicator such as "N/A" in a variable that is otherwise all numbers. In this case, it may be necessary to convert the "N/A" values into something outside the range of expected values for that variable and then label these as "missing" from within the statistical package. It is conventional to use 9, 99, and 999 as missing values as long as these are well outside the expected range of the valid data.

With survey analysis in particular, there are usually instances of values that may be important to some research but not always relevant. For instance, gender may be recorded as: 1 = male, 2 = female, 3 = other, 9 = refused to answer. Most researchers will want only the values for 1 and 2 and should probably use the statistical program to label the values 3 and 9 as "missing." Other researchers will be specifically interested in the individuals whose genders are not clearly identified. In that case they might choose to select only the cases whose values are 3 or 9 in the gender variable, or they could set the values of 1 and 2 to "missing."

2. Naming and labeling

When working with novice researchers who are either creating their own data from original research or synthesizing it from multiple sources, an important role of the data services professional is to encourage users to either create codebooks or to at least use practices consistent with codebook standards. Examples of these practices are creating short variable names that do not include special characters (i.e., \$ or %) as well as longer, more descriptive variable labels, and as-

signing and saving value labels for all non-interval variables. Researchers should be encouraged to do this in code whenever possible, as opposed to doing it via menu-based operations as writing and preserving the code makes the process easily reproducible. In addition, the code can serve as a substitute for a more robust codebook or as a short-list for the researcher, since it is usually interpretable even to someone who is not familiar with the programming language in which it was written.

3. Restructuring

Data users often need to perform some sort of restructuring to a dataset before they begin analysis. Two common forms of restructuring are **subsetting** and **reshaping**. Subsetting involves making the dataset smaller by reducing the number of variables (columns) or the number of observations (rows). Subsetting is a relatively basic operation that can be learned and taught to a user quickly, and it is important because it reinforces fundamental concepts of working with data in a statistical programming language.

Novice users are generally more comfortable thinking of their data in a visual way, but subsetting goes better if the user can think of it from a programmer's perspective. Columns are identified by variable names, rather than by visually locating them and manually doing something to them. The same applies to rows, but in the case of rows there is another level of abstraction. Rows are not identified by name or position, but by characteristics of the variables they contain. So, for instance, a researcher interested in analyzing characteristics of the elderly will most likely want to subset her data to contain only observations about people over a certain age. This type of subsetting (or filtering) is generally done through constructing a logical expression in code, rather than sorting and deleting as one might do in a spreadsheet.

Reshaping involves restructuring the data to match the format required to conduct a particular type of analysis. In the following example from the *World Development Indicators* database (<http://databank.worldbank.org/data/home.aspx>), row subsetting and reshaping are necessary to get data from Table 7.2 (the original format) into an extract where a specific unit of analysis is isolated. In this example *Series.Name* describes the unit of analysis for these data. The goal is to create a subset showing the percentage growth of a country's GDP over time, and the extra variables are dropped. To get there, a preliminary restructuring step brings us to Table 7.3, where the rows are subsetting to contain only the *Series.Name* of "GDP growth (annual %)."

Table 7.2. Initial Data from World Development Indicators

Country.Name	Series.Name	2008	2009
Cyprus	Population density (people per sq. km of land area)	116.5680736	118.0252165
Cyprus	GDP growth (annual %)	3.626616209	-1.666159464
Czech Republic	Population density (people per sq. km of land area)	134.4285178	135.1965825
Czech Republic	GDP growth (annual %)	2.71095557	-4.841785296
Germany	Population density (people per sq. km of land area)	235.5221782	234.9396374
Germany	GDP growth (annual %)	1.052109101	-5.637953129

Table 7.3. First Extract Based on Series.Name

Country.Name	Series.Name	2008	2009
Cyprus	GDP growth (annual %)	3.626616209	-1.666159464
Czech Republic	GDP growth (annual %)	2.71095557	-4.841785296
Germany	GDP growth (annual %)	1.052109101	-5.637953129

In Table 7.3, the data of interest are spread across three columns, each one containing values for a different year. While this may be a convenient form for some purposes, it actually limits the possibilities for analysis. This is because a key attribute of the data, year, is unavailable in this format. In order to remedy this, an operation known as reshaping must be performed. Table 7.4 shows the data in its reshaped format. Notice that there are two new variable names, *year* and *GDP.Growth*.

Table 7.4. Final Extract

Country.Name	Year	GDP.Growth
Cyprus	2008	3.626616209
Czech Republic	2008	2.71095557
Germany	2008	1.052109101
Cyprus	2009	-1.666159464
Czech Republic	2009	-4.841785296
Germany	2009	-5.637953129

In most statistical software packages this type of operation can be performed quickly and efficiently with only a few commands. Figure 7.2 shows the R code used to create Tables 7.2–7.3.

Figure 7.2. R Code Example for Generating Tables 7.2–7.3

```
# load some additional package and the original Table 1 data file
library(dplyr)
library(tidyr)
df <- read.csv("../original_data/Table1.csv")
# filter out the GDP growth rows and display the data frame (Table 2)
df <- filter(df, Series.Name == "GDP growth (annual %)")
# reshape and create Year and GDP.Growth variables
df <- gather(df, Year, GDP.Growth, X2008:X2010)
# clean the leading 'X' from Year
df$Year <- substr(df$Year, 2, 5)
# save Table 3
write.table(df, file = "../generated_data/table3.csv", sep = ",")
```

Getting Started

Given the rise of publicity related to all things data, a host of organizations support individuals motivated to improve their technical skills. In this rapidly evolving landscape it is impossible to create a comprehensive list of resources, but there are three complementary approaches to building and maintaining technical data skills:

- **In-person or online classes.** Data preparation is *not* typically taught in traditional academic courses (e.g., Computer Science or Statistics departments). Fortunately, there exist new online options like Coursera (<https://www.coursera.org/course/getdata>) and School of Data (<http://schoolofdata.org/courses/#IntroDataCleaning>).
- **One-off workshops.** In academia, this is the most common way to teach these skills. If an institution does not offer these types of classes, the non-profit organization Software Carpentry (<https://software-carpentry.org/>) or its affiliate Data Carpentry (<https://datacarpentry.org/>) has an international network of trained instructors available to teach on-site.
- **Self-directed learning.** This works best when working on an actual research project. Countless books exist, but the web is the most up-to-date resource. To start with, here are two highly recommended resource sites: The UCLA Institute for Digital Research and Education (IDRE) (<https://idre.ucla.edu/>) and StackOverflow (<http://stackoverflow.com/>). Resources of this type are growing at a refreshing pace!

Conclusions

*An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.*¹⁵

~ Johnathan Buckheit and David Donoho, Stanford University.

Data service professionals play a key role in assisting and instructing researchers in good reproducible practices. A key element of this argument is that data service professionals engage with researchers at points in the research process where data wrangling assistance is often needed. Despite this, there are obstacles to success, even when the data service professional possesses the right skills to do the job. If the goal of the data services professional is to take on an expanded role, it may take time to develop a reputation as a trusted service provider, particularly when not affiliated with a particular academic discipline. The specific approach should depend on the needs of the institution, in particular on whether other service providers are available. One successful model is to partner with an instructor who grasps the potential benefits of a data service professional's involvement with her students' research projects. Such benefits include the use of more interesting data sources, the application of reproducible research methods, and a higher proportion of instructor time spent with theoretical or methodological questions rather than data manipulation. Another type of "easy win" is to offer workshops on reproducible research for beginning graduate students or undergraduate thesis writers. This approach works well when it catches novices at their points of need.

A potential impediment to success, particularly for those who must make a case for additional training or for investment in software or hardware, is the institutional perception about what is in scope for the data service professional's position. The following points can help make an effective case to administration for investing in a role for data services professionals:

- The fundamental importance of reproducible research to academia
- The benefit of reducing the impact of time-consuming data wrangling questions for the faculty
- The opportunity to instill the sense that providing robust data services is analogous to good citation practice
- Data preparation is not emphasized elsewhere in the curriculum
- Data preparation is a form of information literacy for data.

If on the fence about adding technical data skills to their skill sets, the data services professionals should remember, working with data is "the sexiest job of the 21st century!"¹⁶ But more to the point, data wrangling is a creative endeavor that is both rewarding and an essential component of a robust data service. By

imparting better data manipulation and management skills to our scholars, we empower them as learners and researchers.

1. David Mimno, “Data carpentry,” *David Mimno* (Blog), August 19, 2014, <http://www.mimno.org/articles/carpentry/>
2. Nicholas J. Horton, Benjamin S. Baumer, and Hadley Wickham, “Setting the stage for data science: integration of data management skills in introductory and second courses in statistics,” *arXiv* preprint, February 1, 2015, arXiv:1502.00318.
3. Sean Kandel, “The Sexiest Job of the 21st Century is Tedious, and that Needs to Change,” *Harvard Business Review*, April 1, 2014, <https://hbr.org/2014/04/the-sexiest-job-of-the-21st-century-is-tedious-and-that-needs-to-change/>.
4. Jake Vanderplas. Twitter post.
5. Everett Rogers, *Diffusion of Innovations*, (New York, NY: Simon and Schuster, 2003).
6. The MANTRA research data management training toolkit from has some suggestions on file and folder naming conventions. Research Data MANTRA online course <http://datalib.edina.ac.uk/mantra> by EDINA and Data Library, University of Edinburgh is licensed under a Creative Commons Attribution 4.0 International License.
7. See, for instance, Google’s R Style Guide: <https://google-styleguide.googlecode.com/svn/trunk/Rguide.xml> or Hadley Wickham’s R Packages Style Guide: <http://r-pkgs.had.co.nz/style.html>.
8. Hadley Wickham “Style Guide,” from “Introduction to Data Analysis,” Rice University Fall 2012, accessed on May 18, 2015, <http://stat405.had.co.nz/r-style.html>
9. Scott Long, *The Workflow of Data Analysis Using STATA*, (College Station: Stata Press, 2009).
10. Ben Goldacre, “Scientists Are Hoarding Data And It’s Ruining Medical Research,” last modified 22 July 2015, <http://www.buzzfeed.com/bengoldacre/deworming-trials>.
11. Richard Ball and Norm Medeiros, “Teaching integrity in empirical research: A protocol for documenting data management and analysis,” *Journal of Economic Education*, 43 no. 2 (2012): 182–189.
12. Haverford College, “Project TIER: About,” accessed on May 18, 2015, <http://www.haverford.edu/TIER/about/>.
13. Christine E. Peterson and Deborah Wesley. *The 1993 Indonesian Family Life Survey: Appendix C, Household Codebook, Book I Subfiles (Household Characteristics)*, DRU- 1195/4-NICHD/AID, RAND, 1995, accessed May 18, 2015, <http://www.rand.org/content/dam/rand/pubs/drafts/2007/DRU1195.4.pdf>, 73.
14. For a more detailed examination of codebooks and more example of variable level detail, visit the ICPSR, “What is a codebook,” accessed May 18, 2015, <http://www.icpsr.umich.edu/icpsr-web/NAHDAP/support/faqs/2006/01/what-is-codebook>.
15. Jonathan B. Buckheit and David L. Donoho, “Wavelab and Reproducible Research,” in *Wavelets and Statistics* ed. Anestis Antoniadis and Georges Oppenheim, (New York, NY: Springer, 1995), 59.
16. Thomas H. Davenport and D.J. Patil, “Data Scientist: The Sexiest Job of the 21st Century,” *Harvard Business Review Magazine* 2012, <http://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-Century/ar/1>.