Practical Principles of Data Science for Social and Cultural Analysis

Simon DeDeo, Laboratory for Social Minds, Carnegie Mellon University
http://santafe.edu/~simon

1. **Save alllll the data.** Analysis often begins with data that you scrape, or from an API. When you do so, save every piece of data you can—not just the ones you think you will need later. If you are scraping news articles, for example, for text analysis, you should save not only the publication date and the text of each article, but also the title of the article, the author's name, the section, the date and time at which the article was downloaded, the URL of the article name—and anything else you can find in the page. Don't worry about producing a "standard" output file at this point; just save a Python Pickle (or Ruby Marshal).

2. **Use a simple, robust organization system.** Very often, your data will be a set of objects (e.g., newspaper articles) that have a time ordering (e.g., one article comes before or after another). The objects themselves will have lots of interesting properties. In this case, a natural method is to have a List of Dictionaries (in Python) or equivalently, an Array of Hashes (in ruby).

A Dictionary is a simple way to keep track of an object's properties, because they are accessed via a key; e.g., set[10]["title"] would give you the title of the 11th article. Do not keep multiple arrays for different properties (e.g., title[10] is the 11th title, author[10] is the 11th author). A single off-by-one error will mean your properties won't line up and it is extremely hard to detect.

If your data has a tree structure (e.g., comments in a reddit forum), you can use article ID numbers to keep track. For example, order the comments by date, but in the Dictionary for each object keep a unique ID, the ID of the comment's parent, and a list of IDs of the comment's children.

3. **Do sanity checks.** Plot the number of articles by day. Do you see weird things? For example, do you see no articles on Sunday, or for February? What does the histogram of words/article look like? (Are there no articles with more than 1,000 words? Hundreds with only one word?) How many different authors are there, and who are the most prolific? (Is there one author who appears to have written 90% of the articles?) Spot check the data—do your derived quantities match what you see at the original URL?

4. **Don't use SQL.** SQL is a database language designed for rapid access. It's used on webservers where users expect rapid responses to standard actions; e.g., to see comments in a Wordpress blog and add new ones. The problem with SQL is that the speed requires a highly-structured query system that makes it difficult to pose complex questions. This can even skew the scientific process by discouraging creative lines of investigation. A List/Dictionary structure is slower, but far more flexible. A modern machine can handle Gigabytes of data; it may take five minutes to load in, but you can go for a cup of coffee while you wait.

5. **Process your data, but keep scripts.** For speed, you may want to work with a reduced set. For example, you may just want the number of words in articles over time, so you only need a List of Dictionaries with two keys (date and word-count). Or you may want to process text to remove stopwords. When you create these subsets, keep the scripts you used! Otherwise you will forget whether (e.g.) you included article titles in your count. And you'll be unable to reproduce your analysis from end to end to check for error.