Data cleaning: Tidy data formats

By Radovan Bast, UiT The Arctic University of Norway
ORCID: 0000-0002-7658-1847
Date: April 2021

In data visualisation we often read data from somewhere, we filter the data, post-process it, compute some statistics and finally plot it. And we call this the visualization pipeline. In this video we will discuss how to arrange and store the data in a convenient and robust way for data visualisation pipelines. On this slide I have put an example of storing the data in a spreadsheet. This is an example data set where we observe some arctic species at 3 observation sites on Svalbard (apologies to biologists but I wanted to have a very simple example). Different species are on the left. The columns correspond to observation sites. Note that we also use some colour-coding – red, and blue – which means something (red: multiple observations for the same animal, blue: some problem with the camera). What problems can we anticipate? I recommend you pause the video here and think about how this can be problematic.

One problem could be the format: limited interoperability with other programs. Storing data in spreadsheets can be error prone and there are examples where wrong statistics were computed in high profile papers because the authors did not drag the mouse far enough. More importantly: this particular data layout can be difficult to parse by scripts and thus difficult to automate. Also, it is not in tidy format: difficult to extend (more about this later).

Let us focus on the data structure, not the tool. On the left side I show three examples of arranging the data: top left in a very compact form, below a table where species are arranged in rows, below that a version where species are arranged along columns. Why is this data structure "messy"? How can these 3 examples be problematic for *automated* data visualisation? In the compact representation we would need to somehow divide at the comma. What if we later decide to add more species or more observation sites to the dataset? Then we need to adjust the visualisation pipeline to read additional columns.

In the “tidy data” format, we arrange the data differently: the columns are variables (here observation site and number of sightings), the rows are observations. The order of the rows does not matter. The big advantage of the tidy data format is that we can extend the data with more species and more sites *without* modifying the plotting scripts. This is the recommended form for storing data. However, this does not mean that this is ideal for tables in presentations or publications (more about that in another video).

Which file format should we use to store the data? Here I am listing a number of standard formats: CSV, JSON, GeoJSON, Numpy arrays, HDF5, SQL, and many domain-specific formats. We should use standard formats. Don't invent your own. One of the simplest ones is CSV: comma-separated values. The file consists of a header line and each value is separated by a comma. This is often a good choice. Most visualisation tools can read CSV data.

Often we need to visualise datasets that contain inconsistent or missing entries. Like in this example: note how the date formats are inconsistent and also the university is written in 3 different ways. One row is missing the number of participants. Data cleaning is the process of removing inconsistencies.

Tools such as OpenRefine exist that can help with data cleaning. This does not have to happen manually.

A few words about folder organisation. There is not one right way and the example to the left is only a suggestion. We have arranged the project with subfolders: there is a data folder, a manuscript folder, and a folder for figures. Organise them to be understandable by others and the future you. If you win the lottery today and decide to leave academic research, make sure your research group can still find and understand all related files. This is essential for reproducibility. Add license files to make the data and scripts reusable by others. 'Others' can also be you, in a future job, or in a different research group.

Where to store it? Store the visualisation scripts/notebook and the data in a repository under version control (e.g. GitHub). Get a persistent identifier (DOI) on services like Zenodo or Dataverse. If the data is too big or it is sensitive data, you may need to store it elsewhere, but consider providing a smaller example dataset needed to reproduce plots. Storage options for big data and sensitive data (e.g. patient data) are outside the scope of this video, but I invite you to check our other videos where we discuss these.