Guide for data management

Sarah I. Murphy, et al. (original version was written by Dave Kent & Josh Reichler)

4/10/2020 -

Contents

1 About

This document details recommendations for organizing, moving, and storing data for all members of FSL-MQIP (Food Safety Laboratory and Milk Quality Improvement Program). It is written with the goals of improving the integrity, standardization, and ease of use of the data gathered and used by the lab. The recommendations are not absolute rules, but should be followed unless there is a very good and well-documented reason to deviate from the recommendations. We specifically discuss "projects" throughout this document, however the recommendations apply to many more types of work including grant proposals, research data, manuscripts, sequencing, reports, outside work, etc. Note: If you want to go outside of these recommendations, you need to ensure that your files are accessible to anyone who may need them–this requires you write a plan for accessibilty and have senior lab personnel approve the plan.

2 File Organization

All information pertaining to a project should be kept in the same place, in a single project folder. This includes raw data, molecular work, analysis, notes, and reports, and so on. Exceptions may be made in the case that information is more tightly linked to another project, such as when measurements from one project are used again in another.

2.1 Project Folder Location

It remains a point of discussion where the project folder should be located. We strongly discourage people from saving files to their local computer. Instead, files should be saved either on the server or an online location (e.g., GitHub, Google Drive, Cornell Box). Upon completion of projects all It is agreed that it should be on the lab's network drive, but there are currently two competing practices: inside personal folders, and in top-level folders (such as FOOD-MQIP.) A brief argument for each follows:

2.1.1 Arguments for Personal Folders

• There are closer ties of responsibility between the project and its owner. • Some data might not belong to any one project, and be used by several of one person's projects. • Currently, the more widely used practice.

2.1.2 Arguments for Top-level Folders

• Project information easier to find, since you don't need to remember who owns the project. • If there are multiple people working on a project, they don't need to use each others' folders. • If a project changes hands, you don't need to move all the files or keep your work in someone else's folder.

2.2 The Project README

In each project folder, there should be a file called README. It should contain information on the organization of the project and external data sources. For example, you might specify that the folder Raw Data contains Excel files for Q-Counts and temperature readings copied off of a physical data sheet, and that the statistical analysis you're doing is in the Analysis folder, among other information. The contents of every subfolder in the project should be specified.

Anything that is not immediately obvious about the project's organization from first glance should be detailed. This includes data sources: for every piece of data, if it is not obvious what the source is, it should be documented in the README. If you don't know whether you should document it in the README or not, document it in the README.

2.3 Project Folder Structure

The project folder should have a simple structure, that makes it clear what information and data is kept where. For example:

- 2016_ProjectX
  - raw data
    - q-counts_020816_JR.csv
    - temps_020816_JR.xlsx

- molecular work
  - ...sequencing files...
- analysis
  - stats_021716_JR.Rmd
  - stats_021816_JR.Rmd
- reports
  - weeklyReport-template.doc
  - weeklyReport_020816_JR.pdf
  - weeklyReport_021516_JR.pdf
- notes
  - TODO_021816_JR.txt
  - reminders_021816_JR.txt
  - README_021816_JR.txt
- archive
  - TODO_021716_JR.txt

If you prefer, you an create a folder called "archive" to store older versions of documents (keeping the newest version in the respective folder). One example is provided above for the "TODO" document.

Anything much more complicated than the above example should be detailed in the project README. However, simplicity should be the goal: ideally, even if the README was lost, a person who is unfamiliar with a project should still be able to find any information they're looking for about the project relatively painlessly.

3 General Data Curation

3.1 Raw Data

Original (e.g., Q-count file, piece of paper with coliform Petrifilm counts) or raw data (data that has not been manipulated at all) that is absolutely accepted to be correct, beyond a shadow of a doubt. The integrity of any dataset flows from the integrity of its raw data.

In general, whenever you have a piece of data, it should exist in a single canonical digital form. That is to say, for any given data point, whether it's a row on a spreadsheet, a record in a database, or something else, there should be a single well-known location where the absolutely correct and current version of that data point can be found. If a correction or addition needs to be made to a data set, it should be made to this version, and nowhere else. Each of these canonical data points should be individually dated, so it is known when they were first recorded or last modified.

Files that contain canonical data (i.e. canonical data files) should contain only canon- ical data, and nothing else. They should contain no calculations or analysis, just the raw data. These files may be named whatever is appropriate, but every one that exists in the project folder should be specified in the README. Additionally, the file itself should not be dated, because it contains, by the definition of canonical data, the absolutely correct and perpetually current version of the data.

If a copy of a canonical data file or the data within that file is made, to be used for analysis, reporting, or any other reason, it should be dated and moved away from the original version, so that there is no confusion as to which is the original, and so that you can tell by comparison between the date on the copied file and the dates in the original file if you are working with outdated information. This copied version should not be modified once created. If the canonical data is changed, a new copy should be made. This is to prevent errors where data is added or fixed in one copy, but not another, and having an inconsistent dataset as a result.

3.2 Data Entry

3.2.1 Manual Data

Manually typing in data *should be avoided* when possible. Only manually type in data once, to create its canonical digital version. Simple transcription errors are easy to make and easy to catch, but if left unfixed, they can dramatically affect the integrity of your data. With this in mind, always make sure to double-check your data, especially if it is numeric data. This checking should occur a nontrivial amount of time (at least a few hours) after the initial entry. It is very easy to miss transcription errors that you made right after making them, but do not wait longer than a week to check your data, to prevent having to check hundreds or thousands of entries at a time.

You may use spot-checking to ease the process, if you so choose: check 25% of your data you entered in a particular sitting, and if any errors are found, continue to check the rest of it. However, this cannot be guaranteed to find all errors that exist: if you want to be sure that your data is 100% correct, you must check 100% of your data.

3.2.2 Digital Data

Once a data point is in its canonical digital form, any copying from that data point or deriving information from it (such as separating information kept in a sample ID) should not be done manually. Additionally, any changes or corrections to the data points themselves should be made to the canonical digital version, not to any copied versions, at which point new copies should be made.

This is especially important if you find yourself doing a repetitive process to copy or derive information from many pieces of data. There is almost always a better method you could learn that is easier and less prone to error.1 If you don't know where to start or just want some help figuring out how this might be done, please seek help rather than just doing the task manually.

4 Data Cleaning, Analysis, and Visualization

4.1 Code (programming for statistics)

4.2 Not code

If you are going to make something that is written in code, then you need to document how you did what you did. For example, take detailed notes in a text file while you are cleaning up data or extracting data from a database and then calculating a mean in excel. Another example is that if you use something like OpenRefine, then you can export the project file or the history of what you did and then put that on the server.

5 Version Control

If you are working on something without automated version control, then you need to implement your own plan for version control. For example, everytime that you work on a file, you then rename the file with the date and move the previous version to an Archive folder. For example:

> Sarah opened her manuscript to work on it on 4/16/2020–the last time she had worked on it was 4/15/2020–as such, before she started writing anything new she saved the word doc from previous name of sim4_JDS_manuscript_041520.doc to sim4_JDS_manuscript_041620.doc. Then, she moved sim4_JDS_manuscript_041520.doc to her "Archive" folder.

If you are already working on GitHub then it's already set up for you. You can set-up version control using R, etc.; here are some resources:

- Guides for setting up version control using GitHub & R Studio
  - https://www.molecularecologist.com/2013/11/using-github-with-r-and-rstudio/ (simple guide w/ pictures)
  - https://jennybc.github.io/2014-05-12-ubc/ubc-r/session03_git.html (guide w/ a bit more detail)
  - https://happygitwithr.com/rstudio-git-github.html (guide w/ even more detail)

If you are working Confluence, then version control is also set up for you. Here's a link to setting this up:

Alternatives include Google Drive and Cornell Box, but you must put a copy of these files on the server upon completion of the project.

- If you are working on Google Drive, then here's another link:
- If you are working on Cornell Box, then here's another link:

6 Excel

6.1 Workbook Organization

An Excel workbook contains any number of worksheets. Each worksheet should contain at most one data table, and the worksheet should be clearly named based on the information it contains. If a worksheet contains a data table, it should contain no other information. Notes, metadata, and other information independent of the individual rows should be kept on a different sheet, or outside the file. Charts should be kept on a different sheet from the data they're based on.

If two worksheets contain different versions of the same information, they should be clearly dated or otherwise easily distinguishable.

A project can use any number of workbooks and worksheets within those that is appropriate, but if it's not obviously clear at first glance what data is kept where, it should be detailed in the project's README file.

6.2 Formatting

Formatting in Excel should be used only for the purposes of calling attention to certain phenomena or key pieces of information, and should not be used as a source of informa- tion itself. This is because this kind of information cannot easily be the basis for sorting or filtering, and is lost if the data is exported to formats such as .csv.

Instead of having certain formatting denote something having happened, for example, add a column to the data table to record it. If you'd like to call attention to that event, consider using conditional formatting, which is a great tool for that task, and helps automates the task of formatting application as well.

6.3 Different Kinds of Data

Avoid mixing textual and numeric data in raw data and analysis. This eases the process of working programmatically with your numbers (with Excel and R, for example), and allows you to sort and filter your numeric data more easily. When reporting on the data and analysis, this rule need not be followed, but as long as the data is being used by machines instead of humans, it is easier for the machine to read the information separately.

This means that if you have a data point that includes a number and some meta- information, you should record them in separate columns of your data table. This pertains especially to detection limits with counts. It remains a point of discussion how this kind of information should best be recorded i.e. using <10E vs. 0, but for numbers that you'd ever like to analyze programmatically, it is best by far to keep them separated from any text.

Additionally, avoid keeping manual calculations alongside raw data. If there is a calculation that Excel can do that you would like to include in your table, write an Excel formula to perform it rather than calculating it yourself and entering it manually. This saves time and energy when working with large data sets or correcting older pieces of data.

6.4 Striving for Simplicity

As with project organization, Excel files should be organized as simply as possible. If someone would need a detailed explanation to understand and start to use an Excel workbook, it is probably too complex. In most cases, following the above recommen- dations should accomplish this goal, but always keep it in mind regardless. If you're ever unsure if your Excel workbook(s) is/are too complicated, ask a colleague to find a specific piece of data, and observe.