Module 1 – lessons 03 and 04

Data can be thought of as many different things. We often think of data as numbers or even short text in a spreadsheet. But more often than not, data is “unstructured." Unstructured data includes text, which could come from multiple sources, including not only reports and documents, but books, blogs, and websites. Other kinds of data could be:

* Images and artwork
* video and other media
* interview transcripts
* and any other "RAW" materials needed to complete your project.

Regardless of what kind of “data” you have, your data should be:

* high quality
* reviewed for completeness
* reviewed for mistakes and errors, and
* checked for changes or updates

Because your projects may involve a variety of dynamic data, how do you ensure your reproducible workflow is always efficient? There are several principles to follow. The first starts with organization. Each project you work on should have its own file storage organization structure. Each document, code, script, and product should have a specific purpose, and the versions of these files should all be tracked with a version control system without creating multiple copies of the files.

Following this lecture, I've included a reading page with a helpful example of great organizational structure on Github.

File names should be:

* + - readable by the computer, easy to search, easy to sort (especially by date and author if needed)
    - human readable with logical naming schemes and contain enough info so a human knows what is in the file and what the file is for
    - and short enough to be reasonably manageable
* consider user-based access and security (partitioned by "need to know" *[users with editing and write permissions versus users with read-only access]*

Having an organizational structure for your project is a good idea even if your project only includes yourself, because:

* projects grow
* you may need to support numerous documents and files
* And relationships change and can become complex

No matter what kind of product you want to produce, there should also be instructions on how to use and combine the files in your project. Your documentation is another important component, and it should be clear and well-defined so it can be easily understood by team members at all levels. The documentation could also follow literate programming principles combining the code + text + figures in one document.

Ideally, your final workflow will allow any changes and updates to be automatically incorporated into your final product. You should write code/scripts to automate:

* + - raw data to processed output
    - creating and removing temporary files
    - creating tables, figures and other components
    - assembling the components into final documents, products, and
    - rendering documents into multiple-desired formats

Standardization is also a critical component. Your documentation, code, or templates might be used again in other projects and should be standardized for easier integration and efficiency. You don’t want to reinvent any wheels if you can help it.

Finally, your files, documents, and code should be stored and shared in a centralized way. Cloud-based computing often provides centralized storage and sharing of your projects with your team members and external stakeholders.

Once your project is complete, you should disseminate your work. Why?

* To store and share your data and code. Odds are you will reuse something from this project in a future project.
* To fulfill expectations/requirements to disseminate your findings by the funding agency or publisher of your work
* To increase visibility - when you are listed as the source, you become, by default, THE subject matter expert!
* To increase the speed of collaboration for faster advancement of science and knowledge in your field, and finally
* To increase goodwill with the community and public

Some ways to disseminate your work using Cloud-based solutions are:

* Dropbox
* Google drive
* Github (better with version control and tracking)

Other ways to disseminate may be through:

* Journals - articles, manuscripts
* Books
* Blogs/Websites
* RSS (Rich Site Summary) feeds – like news feeds
* Rpubs – which we will discuss and try out in future lessons in this course
* Other online book platforms such as Gitbook and Bookdown

Some examples of data repositories are:

* GenBank
* PDB

In addition to Github, other data and code sharing repositories include:

* Bitbucket
* Dryad
* Figshare
* Zenodo

A helpful article was published in 2013 in the journal *PLOS Computational Biology* entitled “Ten Simple Rules for Reproducible Computational Research.” While the article focused on applications in computational biology, the key principles they recommended still apply, and include:

* avoid manual steps
* use version control and tracking
* implement standardized formats
* store and track raw data
* organize your output – their list recommends a hierarchical organization
* link textual documentation to the results
* and make the work transparent by allowing public access to scripts, runs, and results

When considering standard practices, think about your own work:

* What do you want to automate?
* What could you re-use?
  + For example, code, files, formatting, graphics, logos, header, footer, boilerplate?
* What should you share with your team?
* What do you find yourself doing over and over?
  + correcting or reformatting?
* If you won the lottery today and left your job, what do you need to tell your replacement so that they can pick up where you left off and complete your current tasks?

The purpose of this course is to help you find the answers to these questions to improve your own workflow, teamwork, and efficiency!

A good example of an organization that follows reproducible principles is 538.com. They write and host stories and opinion pieces covering politics, economics, health, popular culture, and sports. The founder, Nate Silver, and the 538 team are best known for their political polling and forecasting during the United States Presidential and related elections since 2008.

Most of their articles provide references and links to their original data sources, and they also host their data, code, and details behind their analyses on their Github, which is available to the public. We’re going to work with some of these datasets later in this course using the “fivethirtyeight” R package.

It's also worth mentioning Andrew Flowers, one of the contributors to the 'fivethirtyeight' R Package. He gave a great presentation at the 2017 RStudio conference on how to tell stories using data, and he highlighted the various aspects of "data journalism" and importance of workflow, data processing, and transparency in analysis and communication. These are all key aspects of reproducibility.

To really see the power and importance of reproducible workflow principles, let's go back in history to 2001 where an outbreak of a deadly strain of *e.coli* bacteria killed 50 people in Europe. Researchers at the Beijing Genomics Institute worked in collaboration with the Medical Center in Hamburg-Eppendorf to rapidly sequence the genome of the *e.coli* pathogen. Given the severity of the outbreak, the team announced and released the genome via Twitter to the world-wide community of microbial genomicists. A Github repository was established to "crowdsource" analysis and research to find a treatment.

People started contributing their work in under 24 HOURS, and within **5 DAYS** a bacterial agent was proposed to kill the pathogen. This case highlights the importance of these methods and work practices not only for speed and efficiency but also for rapidly addressing problems and developing solutions to save lives.