**Planning for Reproducible Data Science**

**Before Analysis**

* Research Questions / Hypotheses
  + Should have several in mind
  + Should relate to the knowledge gap are you trying to fill
  + Should be coherent and focused
* **Data Acquisition**
  + Choose an experimental design (if appropriate)
  + Consider population structure and confounding
  + Ensure you’ll have data available for validation in the future
  + Balance time and monetary budgets against the number of data points generated or conditions tested
  + Carry out power calculations
  + If limited to a particular data set(s) that don’t provide information needed to answer your research question, you may need to change your research question or hypothesis
* Data Provenance
  + - **Data Source(s)**
      * Record who carried out each experiment, at what date and time, how to contact them
      * Record the model and make information for any equipment used
  + **Data Storage and Management**
    - If you expect your project to include big data, by any definition - volume, velocity, or variety - plan up front for how you’ll manage it
    - If working with big data, consider a distributed file system and parallel processing, as well as streamed processing that can save progress, start, and stop as needed
    - If your data change frequently, include an explicit annotation process, automated if possible, that captures where new files are coming from, at what time, and who’s responsible for them
    - If you’re working with many different kinds of data, consider using a data store such as the Open Science Data Framework that provides detailed typing, metadata, and cross-file integration
  + **Analysis Plan**
    - Consider what could go wrong and how can you avoid it
    - Ensure you have the resources in place (necessary software, sufficient computing power) to carry out the whole project
    - Ensure re-running an entire analysis will not be difficult to carry out
  + **Analysis Infrastructure**
    - Choose a formal scientific workflow environment
    - Find and install software (text editor, development environment, task tracking, documentation, and communication tools) that will facilitate your day-to-day work
    - Set up a revision control repository for the project (GitHub, BitBucket, etc.)
    - Create a directory and folder structure
      * Store files in a consistent and intuitive manner
      * Include a README file that contains information about all other files in your repository

**During Analysis**

* + **Provenance Tracking Infrastructure**
    - Ensure you have infrastructure for at least the following three aspects of your research
      * Analysis commands you can run as part of your main workflow
      * Data that is produced by these, or other, commands
      * Lab notebook for everything else
    - Use a supplemental provenance annotation system and/or environment for formal data provenance
      * JSON, XML, W3C PROV, Open Provenance Model
      * Taverna, Open Science Data, MyGrid
  + **Workflow Documentation**
    - Practice literate programming and extensively document all code
    - Choose easily interpretable variable names
    - Make use of version control
      * GitHub, Google Docs, Etherpad, etc.
    - Make code and workflow as automated as possible
      * Use driver scripts
      * Use platforms like R Markdown, Jupyter notebook, etc.
    - Incorporate positive and negative controls throughout the analysis
    - Consider using a shared file service
      * Dropbox, Google Drive, OneDrive, etc.
  + **Data Quality Control**
    - Document the actions performed to achieve a clean data set
      * Use workflow environments like Doit, Luigi, Taverna, Galaxy, etc.
  + **Verify Analysis Methods**
    - Avoid overfitting using a variety of methods
      * Cross-validation
      * Regularization
      * Bootstrapping

**After Analysis**

* Verify Analysis Results
  + Implement positive and negative results checks
  + Conduct simulation studies
  + Sensitivity analyses
* Implementation Checks
  + Ask at least one person to re-run your analysis and gauge how much effort it takes to implement
    - Include a README file
  + Ensure at least one person knows the basics of what data and methods are stored and how they are documented
  + If possible, re-run your analysis using new and/or different software
* Methods and Documentation for Publication
* **Code Publication**
  + Ensure your code is well documented and commented
  + Consider who will be maintaining your software in the future and how
  + Consider how your code will be available
    - GitHub, BitBucket, etc.
    - As part of the supplementary material for a journal article
    - Your own private web site
* **Data Publication**
  + Cite your data
  + Provide a detailed README file
  + If needed, anonymize or de-identify data
  + Consider who will have access to the data
  + Consider if your data will be available long-term and how (where it will be stored)If providing raw, anonymized or de-identified data is not possible (i.e. proprietary data), provide a synthetic data set that can be used in its place
    - GitHub, Dataverse, Figshare, etc.
    - As part of the supplementary material for a journal article
    - Your own private web site

**Summary of best practices**

**DOs:**

* Start with good science
  + Garbage in, garbage out
  + Coherent, focused questions simplify many problems
  + Working with good collaborators reinforces good practices
  + Something that’s interesting to you will (hopefully) motivate good habits
* **Teach a computer**
  + If something needs to be done as part of your analysis / investigation, try to teach your computer to do it (even if you only need to do it once, like downloading a data set)
  + In order to give your computer instructions, you need to write down exactly what you mean to do and how it should be done
  + Teaching a computer almost guarantees reproducibility
* **Use version control**
  + Slow things down
  + Add changes in small chunks (don't just do one massive commit)
  + Track / tag snapshots; revert to old versions
  + Software like GitHub / BitBucket / SourceForge make it easy to publish results
* **Keep track of your software environment**
  + If you work on a complex project involving many tools / datasets, the software and computing environment can be critical for reproducing your analysis
  + Computer architecture: CPU (Intel, AMD, ARM), GPUs
  + Operating system: Windows, Mac OS, Linux / Unix
  + Software toolchain: Compilers, interpreters, command shell, programming languages (C, Perl, Python, etc.), database backends, data analysis software
  + Supporting software / infrastructure: Libraries, R packages, dependencies
  + External dependencies: Web sites, data repositories, remote databases, software repositories
  + Version numbers: Ideally, for everything (if available)
* **Set your seed**
  + Random number generators generate pseudo-random numbers based on an initial seed (usually a number or set of numbers)
    - **In R you can use the seed() command**
  + Setting the seed allows for the stream of random numbers to be exactly reproducible
  + Whenever you generate random numbers for a non-trivial purpose, always set the seed
* **Think about the entire pipeline**
  + Data analysis is a lengthy process; it is not just tables / figures / reports
  + Raw data → processed data → analysis → report
  + How you got the end is just as important as the end itself
  + The more of the data analysis pipeline you can make reproducible, the better for everyone

**DONT’s:**

* **Do things by hand**
  + Editing spreadsheets of data to “clean it up”Editing tables or figures (e.g. rounding, formatting)
    - **Removing outliers**
    - **QA/QC**
    - **Validating**
  + Downloading data from a web site (clicking links in a web browser)
  + Moving data around your computer; splitting/reformatting data files
  + “We’re just going to do this once ... “
  + Things done by hand need to be precisely documented, and this is much harder than it sounds
* **Point and click**
  + Many data processing / statistical analysis packages have graphical user interfaces (GUIs)
  + GUIs are convenient / intuitive but the actions you take with a GUI can be difficult for others to reproduce
  + Some GUIs produce a log file or script which includes equivalent commands; these can be saved for later examination
  + In general, be careful with data analysis software that is highly interactive; ease of use can sometimes lead to non-reproducible analyses
  + Other interactive software, such as text editors, are usually fine
* **Save output**
  + Avoid saving data analysis output (tables, figures, summaries, processed data, etc.), except perhaps temporarily for efficiency purposes.
  + If a stray output file cannot be easily connected with the means by which it was created, then it is not reproducible.
  + Save the data and code that generated the output, rather than the output itself
  + Intermediate files are okay as long as there is clear documentation of how they were created