Reproducible Data Science

Fundamentals

Reproducibility is a crucial ingredient to ensure that a scientific discovery can be verified or falsified by others. Reproducibility can be thought as an end-to-end process including both wet and dry experiments, i.e. both generation and analysis of data. Let's start with some definitions.

Reproducibility: informally, we mean a range of best practices for quantitative research, including management and sharing of data, and computational methods. More formally, an experiment can be considered reproducible if a different research team can obtain its input data, its computational tools, understand them, and rerun the same methods to obtain the same result.

Replicability: in contrast to reproducibility, replicability means that a different research team can start with the same concept, and perhaps overlapping information, but regenerate their own data and methods that ultimately produce the same result. This is a more challenging and costly process, but this is the scenario where science arguably advances the most. Both reproducibility and replicability rely heavily on a common set of best practices.

Reproducibility of data vs Reproducibility of analysis: the former is usually more about files (where you put them, how you manage them...), and the latter is more about programs, code, and workflows. Roughly, it can be seen as the difference between data and instruction in computer programming.

Controls: positive controls are experimental inputs for which a known result is expected, while negative controls represent null inputs for which no result is expected. Reproducible data science analyses should include positive controls, or unit tests, that consist of defined inputs for which a particular output or result is expected. Similarly, negative controls, in the context of data science, generally represent null unit tests inputs for which no interesting result is expected. If an analysis finds something significant for randomized data, one should be suspicious. Controls, both positive and negative, should always be included and checked automatically at each stage of a study intended to be reproducible.

Electronic vs Protocol reproducibility: electronic reproducibility refers to activities that a computer can carry out automatically. Examples include checking assertions, control results, or unit tests, storing documentation or data in a particular repository, or managing revision control history for an analysis workflow. Protocol reproducibility, instead, refers to things that needs to be carried out by a human, such as proper experiment design, consistent naming schemes, and writing enough documentation.

Data Provenance

Data provenance for reproducible research, can be considered in three phases: the past (*Where did your data come from?*), present (*What did you do with it?*), and future (*Where is it going to live after you publish it?*).
We start by discussing **project design**. This includes best practices to choose and record the origins of data, defining formats, and properly documenting this information. Data provenance can be assisted by electronic aids, such as computational project lay out, scientific workflow systems, electronic lab notebooks, and standards for file naming and recording. Most journals and funding agencies now require some form of public data deposition at the time of manuscript submission, and often an open access requirement, allowing others to freely obtain the data. Towards achieving these goals, one might want to keep in mind the following points:

- Maintain contact information with data generators
- Store well-annotated backed-up copies of the data in standardized locations.
- Understanding the strength and weaknesses of data generating technologies. For instance, for single-cell transcriptomics, one might expect a lot of noise due to very low input material amounts, and a lot of sparsity.
- Maintain electronic records indicating how data have been processed, where the files are, and how each file relates to others.

Scientific workflows are great facilitators. In general, a scientific workflow system captures the provenance of data transformations in a standardized format. This includes some combination of the data products themselves, associated provenance metadata, and the transformations between them (e.g. scripts, executable dependencies, cloud services, web services, or even manual processes in graphical environments). The choice of environments along the spectrum depends on the cost benefit of how much time can be invested into infrastructure versus what the importance or effort of redoing the work later might be. At the lightest-weight end of the workflow spectrum, one can assemble simple workflows in an imperative, non dependency driven, way by centralizing important analysis tasks in a single driver script. In the simplest of cases, the basic rule is that rather than running commands by hand on the command line, writing them into a single driver script is better. A better step up, however, is to use a dependency driven framework as a minimal scientific workflow environment. These will not typically track data provenance explicitly, but they will track data transformations, parameters, and they store all the analyses that have been running, greatly enhancing reproducibility. At the heaviest-weight end of the scale, formal scientific workflow environments typically provide a graphical interface for editing data transformation tasks, import and export of workflow, standardized formats, and task dispatch to a variety of execution engines that might be either local, grid, or cloud based. Common examples in this family include Kepler, Taverna, Arvados, Cromwell, Ergatis, Galaxy, and many others.
We finally discuss data security and privacy. First, the definitions.

- **Privacy**: the need to segregate sensitive data during research.
- **Security**: the need to ensure that only authorized users can access data.

The data necessary to reproduce a study might include sensitive data, such as medical history. Funding agencies typically provide data usage guidelines that balance the reproducibility needs of the research community, with respect for human subjects and protection of research products. In any case, these data must be specially protected by the researcher and shared only with other users authorized for research purposes. A typical technique consists in masking the unique identifiers that could be traced back to an associated human subject. Anonymization of such records is one step that can protect them during research, referring to any method of manipulating the data set that masks these unique identifiers or prevents them from being correctly linked to their associated medical records. Even then, an anonymized data set may still be sensitive enough to require special handling, and the original data set certainly does, leading to the need for data security mechanisms. Data security mechanisms include the physical security of paper documents, limiting researchers or other individuals who have access to the files, and actively preventing social engineering attacks or intrusion attempts. Private data and the security process should not impede reproducibility, although they certainly create additional challenges. Some research can only be carried out using extremely sensitive information, however. Such data sets are typically only released to other researchers under very limited conditions, such as allowing re-analysis only through an entirely network disconnected hard drive or computer. Carried out correctly, best practices for sensitive data handling can allow reproducible research to be executed without risk to participants or others.

Computational Tools

Reproducibility concerns three aspects: code, data, and report generation.

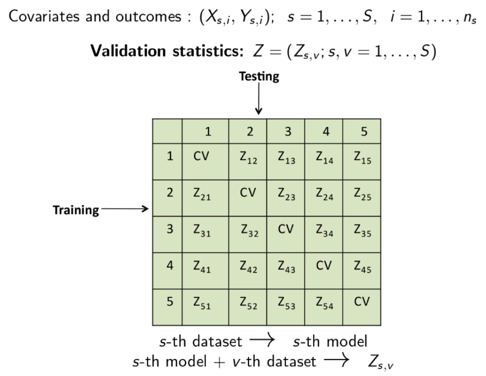
- **Version control:** tools to keep track of changes, edits, and collaborations to code. The primary example is git, alternatives are mercurial and subversion. Features of git allow one to easily track changes to the code among many collaborators. To take the most recent version of code on the server one uses `git pull`, to update the changes one uses `git push` (either using `merge` or `rebase` if in the meantime someone else pushed to the server repository), and to develop more complex features that are tested independently one uses branches.
- **Data management:** crucial questions are *What type of data need to be collected?*, *What is the structure of the data?*, *What format is my community using?*, *Where will the data be stored?*, *How will I share the data?*, *Is there any sensitive information that needs to be protected?* In a word, data should be managed according to a set of best practices known as FAIR: Findable, Accessible, Interoperable, Reusable. In particular, one should document all transformations that have been applied to the data in a notebook or a markdown file. In 2014, a group of researchers at Harvard University published a set of ten rules for the care and feeding of scientific data:

  1. Love your data, and help others love it, too.
  2. Share your data online, with a persistent or permanent identifier.
  3. Conduct science with a particular level of reuse in mind.
  4. Publish workflow as context.
  5. Link your data to your publications as often as possible.
  6. Publish your code (even the small bits).
  7. State how you want to get credit.
  8. Foster and use data repositories.
  9. Reward colleagues who share their data properly.
  10. Be a booster for Data Science.

  Useulf tools include the Harvard Dataverse, the Open Science Framework (OSF), Zenodo, figshare, and Dryad.
- **Literate programming:** programming paradigm introduced by Donald Knuth in which a computer program is given as an explanation of how it works in a natural language, interspersed with snippets of macros and traditional source code, from which compilable source code can be generated. Typical tools used include R Markdowns, knitr (that allow to generate PDF or document file out of the markdown and allows to combine multiple programming languages in the same notebook), and Jupyter notebook.

Statistical Methods

Validating the predicting performance of a Machine Learning model is a fundamental task in Data Science. In this chapter, we study statistical methods for reproducible Data Science. Note that these methods are not just applicable in the classical setting where we have a training and a validation data set sampled from a common distribution, but also when we have multiple heterogeneous data sets as depicted below. When we use different datasets for validation, we use the term cross study validation.



Covariates and outcomes : $(X_{s,i}, Y_{s,i})$; $s = 1, \dots, S$, $i = 1, \dots, n_s$
Validation statistics: $Z = (Z_{s,v}; s, v = 1, \dots, S)$
Testing
Training
1 2 3 4 5
1 CV Z_{12} Z_{13} Z_{14} Z_{15}
2 Z_{21} CV Z_{23} Z_{24} Z_{25}
3 Z_{31} Z_{32} CV Z_{34} Z_{35}
4 Z_{41} Z_{42} Z_{43} CV Z_{45}
5 Z_{51} Z_{52} Z_{53} Z_{54} CV
s-th dataset $\longrightarrow$ s-th model
s-th model + v-th dataset $\longrightarrow$ $Z_{s,v}$

Studying the matrix above can be very informative. In fact, one could cluster the datasets (where the distance between each pair of dataset is given by the corresponding entry in the above matrix), and understand under what condition an algorithm performs well/ poorly.

More generally, we study the following setting. $(X, Y) \sim \mathcal{P}$, and we use a method $M(X)$ to predict $\mathcal{P}$. Note that in general we don't observe $\mathcal{P}$, but we have n i.i.d. samples $(X_1, Y_1), \dots, (X_n, Y_n)$. A measure $Z : (M, \mathcal{P})$ evaluates the prediction quality obtained via M. Note that the crucial assumption to achieve external validity when estimating Z is that the method M has not been trained on the same data used for validation. Some of the most popular validation metrics are summarised below:

- **Coefficient of determination (typically for linear regression models):**
$$R^2 = 1 - \frac{\mathbb{E}\left[(Y - M(X))^2\right]}{Var\left[Y\right]} \in [0, 1]$$
which can be estimated as
$$\hat{R}^2 = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(M(X_i) - Y_i)^2}{\frac{1}{n}(Y_i - \bar{Y})^2} \in [0, 1]$$
- **Brier score (for binary prediction problems):**
$$Z(M, \mathcal{P}) = \mathbb{E}\left[(Y - M(X))^2\right]$$
which can be estimated as
$$\hat{Z}(M, \mathcal{P}) = Z(M, \hat{\mathcal{P}}) = \frac{1}{n}\sum_{i=1}^{n}(M(X_i) - Y_i)^2$$
- **AUROC (for binary prediction problems):** Computed numerically as the area under the receiving operating characteristic curve (ROC). The x-axis on the curve correspond to the FPR, while the y-axis is the TPR. Every point in the ROC curve corresponds to a score for a certain threshold τ. Usually, the different thresholds are picked from the predicted scores (every score correspond to a threshold). For each τ, we classify each score $\le \tau$ to 0, and to 1 otherwise.

Bootstrapping is any test or metric that relies on random sampling with replacement. Bootstrapping allows assigning measures of accuracy (defined in terms of bias, variance, confidence intervals, prediction error or some other such measure) to sample estimates. Bootstrapping is the practice of estimating properties of an estimator (such as its variance) by measuring those properties when sampling from an approximating distribution. One standard choice for an approximating distribution is the empirical distribution function of the observed data. In the case where a set of observations can be assumed to be from an independent and identically distributed population, this can be implemented by constructing a number of resamples with replacement, of the observed dataset (and of equal size to the observed dataset). The typical way to proceed is the following:

1. Replace $\mathcal{P}$ with another distribution $\mathcal{Q} = \hat{\mathcal{P}}$.
2. For B iterations, simulate i.i.d. data from $\mathcal{Q}$ (typically by sampling with replacement from the dataset). Then compute the Z score for the simulated data.
3. Use the B computed scores to make inference about $Z(M, \mathcal{P})$ (e.g. its variance, standard deviation, or confidence intervals).

Survival analysis: a statistical procedure for data analysis in which the outcome variable of interest is the time until an event occurs. Survival analysis has three goals to be addressed:

1. To estimate and interpret survivor and/or, hazard functions from survival data
2. To compare survivor and/or, hazard function
3. To assess the relationship of explanatory variables to survival time

A typical estimator are Kaplan-Meier curves. Let $\tau \ge 0$ be a random variable, which we think of as the time until an event of interest takes place. The goal is to estimate the survival function S underlying τ . This function is defined as
$$S(t) = Pr\left[\tau > t\right]$$
Let $\tau_1, \dots, \tau_n$ be i.i.d. random variables, whose common distribution is that of $\tau : \tau_j$, i.e. the random time when some event j happened. The data available for estimating S is not $(\tau_j)_{j=1,\dots,n}$, but rather the list of pairs $((\tilde{\tau}_j, c_j))_{j=1,\dots,n}$ where for $j \in [n]$, $c_j \ge 0$ is a fixed, deterministic integer, the censoring time of event j and $\tilde{\tau}_j := \min(\tau_j, c_j)$. In particular, the information available about the timing of event j is whether the event happened before the fixed time c_j and if so, then the actual time of the event is also available. The challenge is to estimate $S(t)$ given this data. Note that if $c_k \ge t$, then $Pr\left[\tau_k \ge t\right] = Pr\left[\tilde{\tau}_k \ge t\right]$. This observation yields a natural estimator for $S(t)$:
$$\hat{S}_t = \frac{|k : \tilde{\tau}_k \ge t|}{|k : c_k \ge t|}$$

Patient stratification: in clinical research, stratification has three different meanings.

1. Description of the natural distribution of patients into subgroups, for instance patients may be stratified by stage.
2. Stratification can refer to the process of controlling the random allocation of the treatments in a clinical trial (for instance the randomisation can be stratified on stage) in order to *control for confounders.*
3. Stratification of a trial analysis consists into taking into account patients' characteristics in the analysis.

An example is **risk stratification**. Risk stratification is the process of assigning a health risk status to a patient, and using the patient's risk status to direct and improve care. Said otherwise, risk stratification helps healthcare providers tailor their patient engagement and care management efforts to individual populations with varying healthcare needs.