A reproducible workflow

Gonzalo Rivero

February, 6 2020

Protasis

Why we are here?

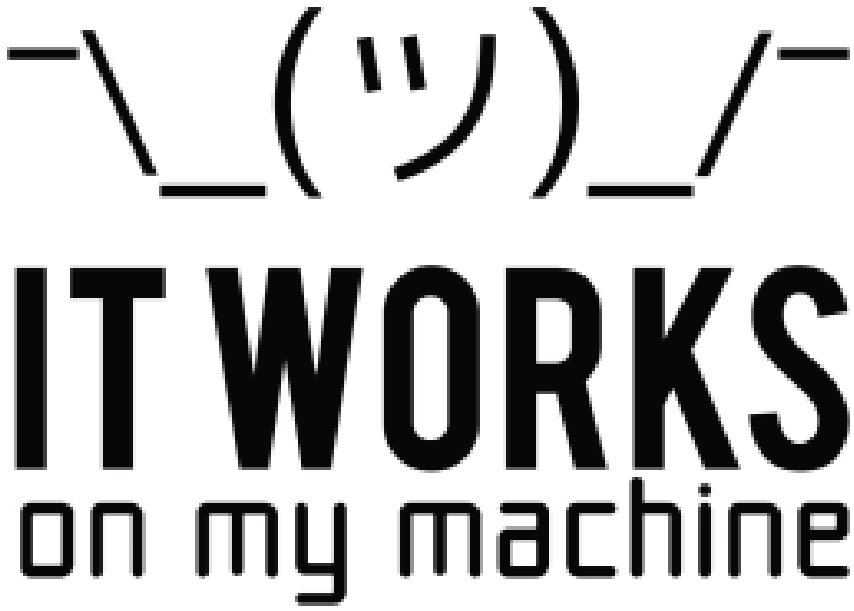

Figure 1: The fundamental problem of reproducibility

The problem (with jargon)

- When we talk about reproducibility we talk about:
  - Idempotence: The same input produces the same outputs.
    - If I run the code twice, I should get the same results both times (there are no hidden states).
  - Conditional independence: The same output is produced regardless of when and where the code is run.
    - If I run the code on a different machine at a later time, I should get the same results (decoupling of computing environment).

Epitasis

The tools we have seen

- *Version control* keeps track of the history of the code. It is often integrated with tools for issue tracking and discussion.
- *Dependency management systems* freeze the dependencies that are needed by the code.
- *Containers* declare the computational environment as code.
- *Pipeline toolkits* specify how each output is generated and only re-run the steps for which the input has changed.
- *Unit tests and assertions* (and code metrics) ensure validity of the code.

How do we put these things together?

Let's rephrase it

- What do we want? We want...
  - ...some assurance that our functions do not contain bugs
  - ...to write code with others
  - ...to run different versions of the code
  - ...the code to always run with the same version of the packages
  - ...the code to always run with the same software
  - ...the code to be executed in a specific sequence
- We may not need to satisfy all these requirements
- We can use some tools to do other stuff

One possible workflow

- All workflows reflect personal tastes about how to piece tools together
- But mine reflects standard best practices :-)
- Overarching ideas:
  - Be predictable
    - Folder structure and file names should be informative
    - Make things easy for humans, not for the machine
  - Everything is code
    - Code is code but text is also code
    - Assumptions should be explicitly declared *in code*
  - A project only needs one command
    - A project has one input and output and only one entry point

Moral

Is that it?

- Reproducibility is good for us
- Reproducilibity *is* automation
  - New tools come out every day and constant learning is unavoidable
  - But principles are (more or less) stable
  - Science *is* code. Get comfortable with it.
- Reproducibility is a collaborative process
  - It is a form of communication with colleagues and clients
  - It is challenging. It can't be done without help