Reproducibility of computational workflows is automated using continuous analysis

Brett K Beaulieu-Jones[1] & Casey S Greene[2]

Replication, validation and extension of experiments are crucial for scientific progress. Computational experiments are scriptable and should be easy to reproduce. However, computational analyses are designed and run in a specific computing environment, which may be difficult or impossible to match using written instructions. We report the development of continuous analysis, a workflow that enables reproducible computational analyses. Continuous analysis combines Docker, a container technology akin to virtual machines, with continuous integration, a software development technique, to automatically rerun a computational analysis whenever updates or improvements are made to source code or data. This enables researchers to reproduce results without contacting the study authors. Continuous analysis allows reviewers, editors or readers to verify reproducibility without manually downloading and rerunning code and can provide an audit trail for analyses of data that cannot be shared.

Leading scientific journals have highlighted a need for improved reproducibility to increase confidence in results and reduce the number of retractions[1–5]. In a recent survey, 90% of researchers acknowledged that there 'is a reproducibility crisis'[6]. Computational reproducibility is the ability to exactly reproduce results given the same data, as opposed to replication, which requires an independent experiment. Computational protocols used for research should be readily reproducible, because all of the steps are scripted into a only with discrepancies. Additionally, Hothorn and Leisch found that more than 80% of manuscripts did not report software versions[10].

It has been proposed that open science could aid reproducibility[3,11]. In open science, the data and source code are shared. Intermediate results and project planning are sometimes also shared (as, for example, with Thinklab, https://thinklab.com/). Sharing data and source code is necessary, but not sufficient, to make research reproducible. Even when code and data are shared, variability in computing environments, operating systems and the software versions used during the original analysis make it difficult to reproduce results. It is common to use one or more software libraries during a project. Using these libraries creates a dependency on a particular version of the library; research code often works only with old versions of these libraries[12]. Developers of newer versions may rename functions, resulting in broken code, or change the way a function works to yield a slightly different result without returning an error. For example, Python 2 would perform integer division by default, so 5/2 would return 2. Python 3 performs floating-point division by default, so the same 5/2 command now returns 2.5. In addition, old or broken dependencies can mean that it is not possible for readers or reviewers to recreate the computational environment used by the authors of a study. In this case it becomes impossible to validate or extend results.

We first illustrate, using a practical example, the problem of reproducibility of computational studies. Then we describe the development and validation of a method named continuous analysis that can address this problem.