Embedding Queryable Provenance in Research Objects to Achieve Research Transparency

Timothy McPhillips Lan Li Nikolaus Parulian Bertram Ludäscher
School of Information Sciences, University of Illinois at Urbana-Champaign
{tmcphill,lanl2,nnp2,ludaesch}@illinois.edu

1 INTRODUCTION

Challenges of reproducibility

Challenges of terminology: Reproduce vs Replicate The difference between them Standardization proposals History of usage in biology Cell and molecular biologists study both replication and reproduction as natural processes. DNA replicates (high fidelity, variation not desired, ingredients indistiguishable). Organisms reproduce (lower fidelity, variation desired, different ingredients). Cells have replisomes, complex molecular machines where DNA replication occurs, and copying errors are detected and corrected. In origin of life research a crucial debate is over 'replication first' (DNA World) or 'metabolism first' (aka reproduction first, i.e. without replicating genetic material). These terminologies are well established. Biologists also have a rich and well-defined vocabulary to describe replicability of experimental measurements and results. Commonly distinguish two kinds of experimental 'replicates': technical replicates, and biological replicates. FASEB definitions of reproducibility and replicability are consistent with the above. Footnote: provide number of FASEB members, list subdisciplines represented by FASEB. Reality is that the terminology is well established in large branches of science already. Need for reproduction/replication to mean different things in different fields The relationships of corresponding concepts across fields is one of analogy, not identity. Exact repeatability is at least theoretically possible and sometimes practical under realistic assumptions when an experiment is entirely in silico and isolated from the outside world. As soon as observation of the real world is involved, exact repeatability often is impossible. Neither of the types of replicates in experimental biology are exact, although both are measures of repeatability. There often is no way to repeat exactly an experiment that involves scientific instruments, physical samples, or experimental apparatus. In contrast, it is not unreasonable to talk about exactly repeating a purely computational experiment, at least by the original researcher, on the same hardware, close in time to the original experiment. In reality, computational repeatability is not as easy as sometimes assumed other under conditions, but fundamentally this is a different situation than when a scientific instrument is involved or observations are made of the external world. Our recommendation on terminology Respect differences between fields of research and different expectations with regard to reproducibility. Use the R* words in ways that make their meanings clear in context. Do not be surprised if computational sciences not representative of science generally. Specific implications for Research Objects Take care to define R* words. Do not expect efforts to achieve computational reproducibility to enable reproduce science generally.

Challenges of computational reproducibility The fundamental limitations of computers are well understood. Finite precision arithmetic, different word sizes on different processors, round-off errors impose limits on scientific computations and their repeatability across different computing environments. Virtual machines and containers do not address these issues. Full emulation is required to run the same binary in identical fashion on a different processor, and this is typically slow. These limitations are even more challenging to manage reproducibly because programs typically are compiled, meaning that the exact sequence of machine instructions executed even by a single processor cannot generally be controlled. A different compiler, or a newer version of the same compiler will yield a different sequence of machine instructions. Yet this is the easiest kind of computational reproducibility to talk about and achieve. Reproducing the software running the program is particularly challenging in practice. How can we ensure that the same 'bitstream' for two executions is identical? Even holding the computer hardware and compiler version constant, programs depend on language libraries, OS libraries, and system calls. Much scientific software also depends directly and (and even more) indirectly on large numbers of 3rd-party libraries. Only direct dependencies can be controlled reliably. And many dependencies are via shared libraries that can change between executions of the exact same executable. Recompiling or even just rerunning the "same" program a week later can result in a completely different effective instruction stream.