Popper: Creating Reproducible Computational and Data Science Experimentation Pipelines

Date: June 13, 2018
Presented by: Ivo Jimenez of UC Santa Cruz
homepage: github.com/systemslab/popper
documentation: popper.rtfd.io/en/docs-reorg

Please type your questions in the document below and we will address your question during the presentation.

Q:What if datasets are large? They can probably not be captured. For example I have to work on large data of hundreds GB, how would this be integrated into popper?

A: For datasets that are large enough that they cannot be managed by Git, solutions such as a PFS, GitLFS, Datapackages, ckan, among others exist. These tools and services allow users to manage large datasets and version-control them. From the point of view of Popper, this is just another tool that will get invoked as part of the execution of a pipeline. As part of our documentation, we have examples on how to use datapackages, and another on how to use data.world.

Q: ACM replicability: I would argue that to really believe that a 3rd party can replicate the process, you probably need to run the pipeline under a different set of user credentials in order to have confidence that there is not something hidden in your environment. Comments?

A: Totally agree. We have found instances in practice where an experiment would not run on another user's account because the experiment scripts assumed a certain length for the string that holds the username. The practice we follow to make pipelines portable is to parametrize user credentials (in the form environment variables), in such a way that it easily allow other users to re-run on their environment

---

Q: Do you have experience yet using this approach in computational science research, as opposed to computer science?

A: Yes, the goal for Popper is to make it a domain-agnostic experimentation protocol. Examples of how to follow Popper on distinct domains: atmospheric science, computational neuroscience, genomics and applied math. We recently also documented how to apply the Popper protocol for long-running computational experiments.

Q: What is the best way for interested users to interact with you?

A: We have a Gitter channel where we are always available. We also use Github issues to address any questions (for example this one)

Q: How do you check which parts of the pipeline should be re-executed?

A: A popper pipeline is just a list of bash scripts that get invoked sequentially. When they get executed, the popper command checks their exit code and immediately stops if non-zero is returned. If the pipeline gets re-executed, all the stages get re-executed unless the --skip flag is passed to the

---

Q. Regarding the integration with minting & archival services such as zenodo, all the data, recipes and environments (via containers) are copied into the archival service? This is how one searches for and finds the digital data artifacts from the scientific work in order to reproduce it?

A: The ‘popper archive’ command compresses (in a zip file) the files in the repository. This does not include artifacts that are generated by the scripts in the repository such as compiled code (binary files), container or VM images, output datasets on remote file systems, etc.

The ‘popper search’ command looks for keywords on the README associated to a pipeline. For example,

---

Q. As you discussed, popper is not applicable to dynamic workflows, correct?

A: Yes, a Popper pipeline is a simple sequence of bash scripts. Popper is not a replacement for scientific workflow engines, instead, its goal is to capture the highest-most workflow: the human interaction with a terminal. For more on this, please take a look at the Popper vs. other software section of our documentation.
