CENTER FOR RESEARCH IN
OPEN SOURCE SOFTWARE

Reproducible Computational Science with Popper Workflows

CROSS 2020 Research Symposium

Anders Poirel

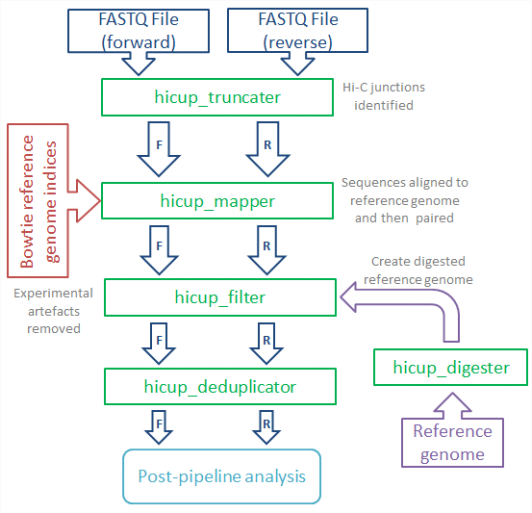
University of California, Santa Cruz

Common Reproducibility Issues (1)

Python statistics libraries

- Research using statistical and/or machine learning tools depends on increasingly complex software stacks
- Replicating a result is difficult due to incomplete dependencies, platform incompatibilities, ...

Common Reproducibility Issues (2)



A bioinformatics data pipeline

- Data processing pipelines have many interdependant steps from the raw data to the final result
- Replicating a workflow requires difficult guesswork if these steps not properly documented

Popper

Popper is a container-native workflow execution engine which addresses these issues:

- Containerizing workflows avoids problems due to different software environments
- Specifying steps explicitely in a Popper workflow file avoids problems due to data pipeline complexity

Limitations

However . . .

- Popper requires fluency in DevOps tools (in particular container engines)
- Steep learning curve for users with no prior experience in these tools, who would nonetheless benefit from more reproducible workflows

→ My goal was to create a collection of tools and tutorials to smooth out this learning curve for users in computational fields

Contributions: Tutorials

Tutorial for developing Popper workflows oriented to Python and R users:

- Computational notebooks with Popper
- Dependency management
- Project structure

Contributions: Templates

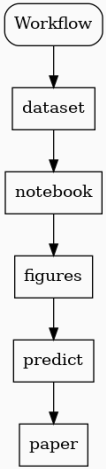
cookiecutter templates to bootstrap Popper workflows in Python and R including starter Docker images

```
├── LICENSE
├── README.md                    <- The top-level README.
├── data                         <- Data used in workflow.
├── paper                        <- Generated paper as PDF, LaTeX.
├── wf.yml                       <- Workflow starter
├── containers                   <- Definitions of containers used in workflow
│   └── exploration              <- Container used for exploratory work.
│       ├── exploration.dockerfile  <- Default dockerfile used in workflow.
│       └── exploration_env.yml  <- Defines conda environment used by container.
├── results
│   ├── models                   <- Model predictions, serialized models, etc.
│   └── figures                  <- Graphics created during workflow.
└── src                          <- Source code for this project.
    ├── notebooks                <- Jupyter notebooks.
    ├── data                     <- Scripts to download or generate data.
    ├── models                   <- Scripts used to generate models.
    └── figures                  <- Scripts to generate graphics.
```

Structure generated by cookiecutter for Python workflows

CROSS CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

Contributions: Sample Workflows



Workflow diagram generated by Popper

Sample Popper workflows in Python and R demonstrating an end-to-end machine learning project with data acquisition and transformation, model fitting and evaluation

Highlights (1)

- **Key idea in DevOps**: minimize the distance between development and production environments
- Users in computational research should be able to develop from the beginning their workflows in Popper
- This is done using Popper's interactive execution feature

Highlights (2)

- Users in computational fields use computational notebooks to prototype workflows
- I added configuration options to the core popper CLI to support running a computational notebook server

Thank you!

I'd like to thank Ivo Jimenez for his mentorship this summer!

Find my work on this project at https://github.com/getpopper

- Tutorial for getting started is under "guides" in the documentation
- Use the repositories labelled "cookiecutter" to bootstrap a workflow