A workflow for continuous and collaborative benchmarking

This manuscript (permalink) was automatically generated from komparo/manuscript-workflow@cb72fab on October 25, 2018.

Authors

- **Wouter Saelens**
  0000-0002-7114-6248 · zouter · zouters
  Ghent University · Funded by Fonds Wetenschappelijk Onderzoek
- **Robrecht Cannoodt**
- **Lukas Weber**
- **Charlotte Soneson**
- **Yvan Saeys**
- **Mark D. Robinson**

Abstract

Benchmarking is a critical step in the development of bioinformatics tools, but the way it is done at the moment has some limitations. Because each benchmark is developed in isolation, they tend to be hard to compare, extend and are rapidly outdated. To address this, we combined modern software development tools to create a workflow for continuous and collaborative benchmarking. The structure of the benchmark is highly modular, so that anyone can contribute a set of datasets, metrics, methods or interprete the results, and get credit for their input. We apply this worklow on two emerging types of methods in the single-cell analysis field: trajectory differential expression (https://github.com/komparo/tde) and trajectory alignment (https://github.com/komparo/ta).

Introduction

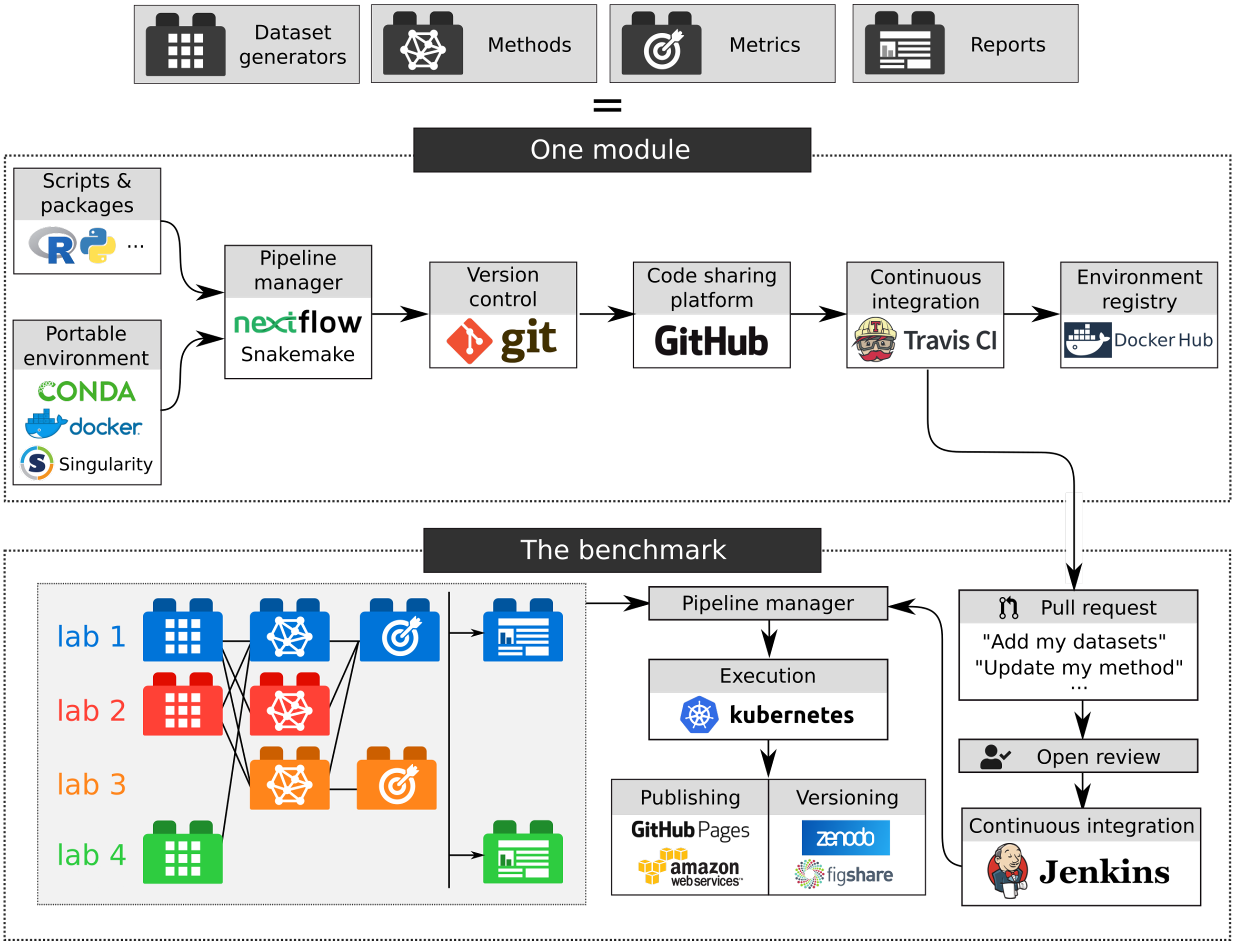

Figure 1: The pipeline.

Modules

Our pipeline consists of several “modules”, which are integrated and connected to ultimately produce a full benchmarking pipeline. These modules contain the code to generate some datasets, run a method and compare the output. This is similar to some related work (SummarizedBenchmark [1] and Dynamic Statistical Comparisons, https://github.com/stephenslab/dsc). Moreover, to assure a balanced interpretation of the results, report modules will aggregate and summarise the results, and provide some interpretation.

A module in our proposed workflow is more than just code, but includes several

Scripts and packages

There large diversity in programming languages used in bio-informatics, even within particular a particular subfield (such as single-cell bioinformatics [2]). A collaborative workflow should therefore never "lock-in" to a particular language,

Portable environment

Pipeline manager

- Controls reproducible execution of the scripts within the environment
- Input and output should always be explicitely defined
- Rerunning the module, or parts of the module, should only be triggered if input has changed

Version control

- Crucial for keeping track of what was changed when
- Also crucial for working together

Code sharing platform

Code sharing is more than a place to deposit code: - Create issues - Create pull requests - Versioning the code

Automated testing and continuous integration

- 90% of the errors are caught here

Environment registry

- Easily downloadable by anyone wanting to replicate the environment

Combining modules within a benchmark

Combining modules

Pipeline manager

Execution

Continuous publishing

Adding or updating a module

Continuous integration

Versioning

Outlook

The project as it stands now is meant to be a proof-of-concept. In the ideal case, a continuous benchmarking project should be supported by a larger consortium, such as the Human Cell Atlas, which would not only assure its continuity, but would also provide infrastructure support. Perhaps the code platform, continuous integration, and environment registry servers could be hosted by such as consortium. By providing a platform where old ideas are rigorously tested, and new ideas can be easily validated,

A platform like this should be build upon the idea that future methods and output formats can never be predicted, but at least we can prepare for them.

References

1. **Reproducible and replicable comparisons using SummarizedBenchmark**
Patrick K Kimes, Alejandro Reyes
Bioinformatics (2018-07-17) https://doi.org/gdvt5p
DOI: 10.1093/bioinformatics/bty627 · PMID: 30016409

2. **Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database**
Luke Zappia, Belinda Phipson, Alicia Oshlack
PLOS Computational Biology (2018-06-25) https://doi.org/gdqcjz
DOI: 10.1371/journal.pcbi.1006245 · PMID: 29939984 · PMCID: PMC6034903