A workflow for continuous and collaborative benchmarking

This manuscript (permalink) was automatically generated from komparo/manuscript-workflow@bbee2d5 on November 17, 2018.

Authors

- **Wouter Saelens** *
  0000-0002-7114-6248 · zouter · zouters
  Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium · Funded by Fonds Wetenschappelijk Onderzoek

- **Robrecht Cannoodt** *
  · rcannood · rcannood
  Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium · Funded by Fonds Wetenschappelijk Onderzoek

- **Lukas Weber**
  · lmwebr
  Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland; SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland

- **Charlotte Soneson**
  · CSoneson
  Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland; SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland; Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

- **Yvan Saeys** *

  Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

- **Mark D. Robinson** *

Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland; SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland

Abstract

Benchmarking is a critical step in the development of bioinformatics tools, but the way benchmarking 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 Because each benchmark is developed in isolation, they tend to be hard to compare, extend and are rapidly outdated. Moreover, benchmarks are usually rapidily outdated as new methods are developed. To address these challenges, 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 contributions. We apply this worklow on an emerging type of analysis in the single-cell field: trajectory differential expression, available at https://github.com/komparo/tde. A skeleton version of the workflow, which can be used to create a similar benchmarking workflow for a different type of methods, can be found at https://github.com/komparo/skeleton.

Introduction

Evaluating the performance of a new method, and comparing it to the state-of-the-art, is a critical step in the development of bioinformatics methods. Benchmarks are essential to showcase the advantages and weaknesses of a method, and assure that new tools improve upon related methods. Despite this, well-designed and balanced benchmarking strategy can be difficult to create, especially when a ground truth on real data is not available.

The breadth of a benchmark is influenced by its purpose. In some studies, the goal is to review the methods available in the field, and highlight current challenges. Such independent benchmarks are usually very comprehensive, involving many datasets and different metrics ranging assessing the accuracy, scalability and robustness of a method. A special case of such a benchmark are competitions, where the focus lies on promoting the development of new methods within the field, while using existing methods as baseline. Other benchmarks are used as a companion to a study proposing a new method, demonstrating its improvements and usefulness.

While benchmarks are unmistakingly important, the way benchmarking is usually done has some limitations:

- Benchmarks are quickly outdated when new methods come along.
- Benchmarks are difficult to extend, as this is usually only added as an afterthought.
- While benchmarks often reach different conclusions, they are difficult to compare, because of (unclear) differences in datasets, method parameters, metric implementation and aggregation.
- Independent benchmarks and competitions tend to be authoritative, with only a small group of people deciding on how methods should be compared.
- Independent benchmarks are usually published quite late, only after a lot of methods are already available.

- Companion benchmarks represent in some way a lot of wasted effort, because datasets are often reanalysed, metrics reimplemented, and methods rewrapped.

To resolve these issues, we created a workflow for benchmarking which centers around the following three core concepts:

- **Modular**: It should be possible to extend the benchmark simply by adding a self-contained "module". Such a module could be: a dataset generator, a method, a set of metrics, or a report generator that interpretes the metrics and produces a report. Several tools exist already for making benchmarks modular: SummarizedBenchmark [1], Dynamic Statistical Comparisons and iCOBRA [2].
- **Collaborative**: Anyone with a computer and internet connection should be able to run and contribute to the benchmark. This can range from contributing a module, to changing the structure of the benchmark itself. Discussions on the benchmark or any of the reports should also be open. The collaborative aspect of benchmarking has usually focused on the level of methods, with countless competitions and challenges, such as those organised by DREAM or kaggle.
- **Continuous**: A benchmark should be continously updated when new modules are added. This has quite a long history in bioinformatics, particularly in structure prediction [3], but also in other fields [4].

To construct a workflow which fulfills combines these three concepts, we used several ideas and tools coming from modern software development, such as continuous integration, containerisation and workflow management.

The overall structure uses several different **types of modules**: dataset generators can generate datasets and optionally use another dataset as input, methods use a dataset to generate some model, metrics will calculate some scores using the model and optionally also parts of the dataset, and finally a report generator which summarise the datasets, models and scores into a report. Each type of module can generate a set of files which are constrained to a particular set of **formats**. Each format has an unambiguous description, a set of good and bad examples, and includes a validator which validates the output files generated by each module. While each format is defined beforehand, new formats can be added over time as the field progresses. A **module** is a set of scripts and packages, which are run inside a portable environment. This module is put under version control, shared on a code sharing platform, and tested automatically using continuous integration. When all tests of a module are succesful, these modules can be integrated into the actual **benchmarking workflow**. Within this workflow, modules are connected through a particular design, which is executed using a workflow manager. The output of the benchmark are a set of reports and apps, which are made available through a publishing platforms. To add a new module, a pull request is created to integrate the module within the benchmarking workflow, after which the contribution is reviewed openly. When accepted, the module is automatically integrated within the

workflow, and the necessary parts of the workflow are re-executed. Finally, in regular time intervals (e.g. monthly), the full set of reports and apps are gathered and versioned.

As a test case, we developed a proof-of-concept benchmark for single-cell trajectory differential expression (TDE) methods. TDE methods try to find genes which are differentially expressed along a trajectory, the latter of which is an positioning of cells along a graph structure. Given that only a few of such methods have been developed yet [5,6,7], this is the ideal scenario for developing the idea of a continuous and collaborative benchmark, and try to find solutions to the inevitable challenges which will come up as the field develops.

We will further discuss each element of the workflow in detail, along with how we currently implemented it in practice. It is important to acknowledge here that this is only one possible implementation, and that other tools, some of which still have to be developed, could better fit the benchmarking usecase. In the end, what is the most important are not the way a benchmark is implemented, but the ideas behind its implementation.

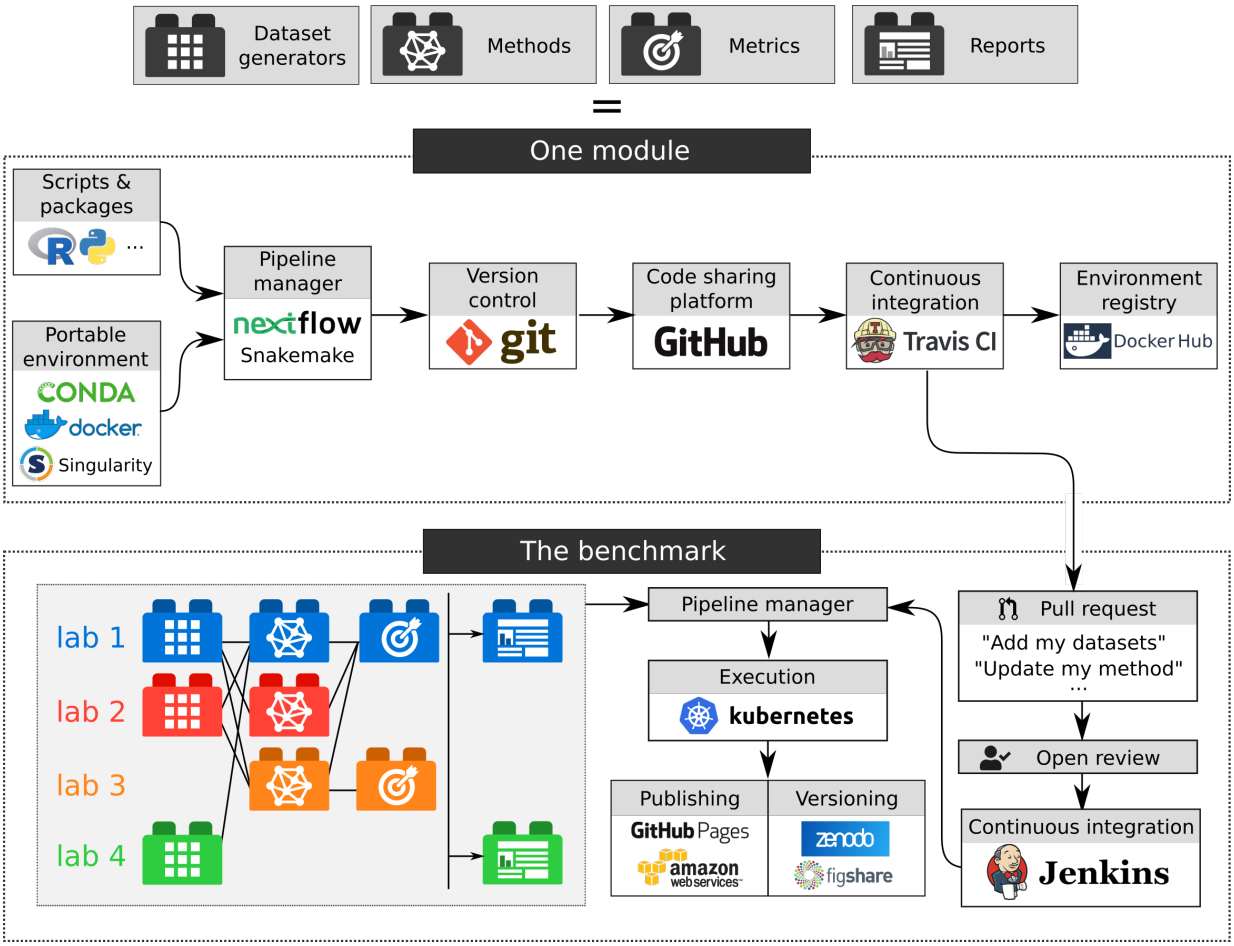

Figure 1: The pipeline.

Data formats

The basis of any collaborative benchmark are A common format to exchange data between modules is critical, but also challenging. Sometimes, the differences can be minor, for example whether the samples within a gene expression matrix are put in the rows or in the column. In other cases, different data formats can have a significant impact on storage and/or the speed by which the data can be processed.

- Common formats should be clearly defined, so that someone that develops a new module knows what the input will look like, even if it is generated by others
- Formats should be validated continuously throughout the benchmark, only modules which adhere to the formats should be included
- Formats should not be dictated by one “commitee”, instead, anyone should be easy to add a new format although it is usually recommended to use the existing formats
- Formats should be able to change as the field progresses, given that it is impossible to know what the optimal format will look like at the start
- Having at least one common format, even if it is suboptimal for certain use cases, is better than having none at all.

Defining a format: - Several good and bad examples - Validation, in R or existing tools (JSON schema)

How can formats be **dynamic**: - Extensions: New fields in a json file, or new columns in a csv file. Because this would not invalidate old formats, this can be easily added. - Additions: New formats are added next to the old. For example, a new sparse expression format, next to the old “dense” expression format. Old modules won’t work with these new datasets, unless: - The module also outputs the old format - We write convertors between formats, which are automatically called after some module is finished. - Replacements: Replacing old formats requires some “versioning” similar as what is done elsewhere (eg. JSON schema, web APIs, HTML/XML versions, …). Here also, convertors would be necessary for interopability, so that old modules still work. Alternatively, these old modules could be deprecated until they conform to the new format.

Module types

Modules

Our pipeline consists of several “modules”, which are integrated and connected to ultimately produce a full crowd-sourced benchmarking pipeline. These modules contain the code to generate some datasets, run a method and compare the output. Moreover, to assure a balanced

interpretation of the results, report modules will aggregate and summarise the results, and provide some interpretation.

While the idea of creating a more modular benchmarking workflow is not new (SummarizedBenchmark [1] and Dynamic Statistical Comparisons, https://github.com/stephenslab/dsc)

A module also contains several other components which are in our view necessary to make the workflow easily extendable.

Scripts and packages

A very varied set of programming languages used in bioinformatics, even within particular a particular subfield (such as single-cell bioinformatics [8]). A collaborative workflow should therefore avoid a “lock-in” to a particular language.

Portable environment

Pipeline manager

- Provides an interface between different modules
- 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
- Checks whether the inputs are present
- Checks whether the outputs are created and validates this output

Version control

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

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

Testing a module: - Checks the modules content, e.g. if the metadata is complete - Checks whether it fullfills the requirement for this module, e.g. if it wil generate the required outputs - Tests whether it can be loaded - Tests whether it can be run using small input data - Validates the produced output

While continuous integration for every module can sound like overdoing it, 90% of the errors are caught here. For small benchmarks, it is overkill, for large benchmarks, it is indispensible for maintainability

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. Technologies, and the companies and communities building them, come and go, and the tools we used for this benchmark will almost certainly feel outdated in a couple of years. The crucial point is not which tools are used, but what advantages they provide for the community: a portable environment, a reproducible workflow, a way to collaboratively design a benchmark, and ultimately a more democratic view of the field and its challenges lying ahead.

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. In particular, services which have strong requirements on the side of storage and/or computing power would benefit from this, such as continuous integration, the environment registry, and the execution cluster.

By providing a shared platform where old and current 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.

Reports as a forum

- Discuss multiple possible interpretations
- Self-assessment trap [9]

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. **iCOBRA: open, reproducible, standardized and live method benchmarking**
Charlotte Soneson, Mark D Robinson
Nature Methods (2016-04) https://doi.org/gfj2zx
DOI: 10.1038/nmeth.3805 · PMID: 27027585

3. **Critical assessment of methods of protein structure prediction (CASP)-Round XII**
John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, Anna Tramontano
Proteins: Structure, Function, and Bioinformatics (2017-12-15) https://doi.org/gfj2zw
DOI: 10.1002/prot.25415 · PMID: 29082672 · PMCID: PMC5897042

4. **A benchmark for RNA-seq quantification pipelines**
Mingxiang Teng, Michael I. Love, Carrie A. Davis, Sarah Djebali, Alexander Dobin, Brenton R. Graveley, Sheng Li, Christopher E. Mason, Sara Olson, Dmitri Pervouchine, … Rafael A. Irizarry
Genome Biology (2016-04-23) https://doi.org/gfj2zz
DOI: 10.1186/s13059-016-0940-1 · PMID: 27107712 · PMCID: PMC4842274

5. **A descriptive marker gene approach to single-cell pseudotime inference**
Kieran R. Campbell, Christopher Yau
Cold Spring Harbor Laboratory (2016-06-23) https://doi.org/gfj23j
DOI: 10.1101/060442

6. **SCORPIUS improves trajectory inference and identifies novel modules in dendritic cell development**
Robrecht Cannoodt, Wouter Saelens, Dorine Sichien, Simon Tavernier, Sophie Janssens, Martin Guilliams, Bart N Lambrecht, Katleen De Preter, Yvan Saeys
Cold Spring Harbor Laboratory (2016-10-06) https://doi.org/gfj23n
DOI: 10.1101/079509

7. **Reversed graph embedding resolves complex single-cell trajectories**
Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, Cole Trapnell
Nature Methods (2017-08-21) https://doi.org/gc5v2g
DOI: 10.1038/nmeth.4402 · PMID: 28825705 · PMCID: PMC5764547

8. **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

9. **The self-assessment trap: can we all be better than average?**
R. Norel, J. J. Rice, G. Stolovitzky
Molecular Systems Biology (2014-04-16) https://doi.org/bxxmvz
DOI: 10.1038/msb.2011.70 · PMID: 21988833 · PMCID: PMC3261704