Dear Editor,

Thank you for handling our manuscript. Please find below a point-to-point response to the reviewers' comments, as well as the highlighted and clean and updated manuscript versions.

Thank you in advance for processing our submission.

Best regards,

Mark D. Robinson and Izaskun Mallona

University of Zurich

Reviewer #1

Reviewer #1: The proposed topic of this piece -- continuous is of vast potential interest. The ongoing proliferation in methodologies creates an urgent need for enhanced and automated benchmarking. Unfortunately the text itself largely covers extremely well-trod terrain.

First, it overviews benchmarking, with some nods to best practices. This has been done dozens of times. A very old example that covers most of the same topics is: https://www.embopress.org/doi/full/10.1038/msb.2011.70 But of course there are dozens of more recent examples. Topics like parameterization, method-sensitivity to parameterization, the need for reporting, their incorporation into benchmarking approaches and so forth have been widely covered. The text on benchmarking as a whole does not synthesize or provide a new, clear description, but rather reads as a standard summary of a well-understood field.

We agree that there is a need for enhanced and automated benchmarking. In order to differentiate the focus of our discussion, we have added a new Figure 1 and extended the introduction to better clarify the manuscript scope. In particular, we now make more explicit the distinction between the current practices in scientific benchmarking and benchmarking automation via benchmarking systems (such as OpenEBench, openproblems.bio, etc.), which was not clear enough.

Second, the authors make a call for "benchmarking systems", but it is extremely unclear what this means. Many challenges have standard input/output/scoring/analysis, and indeed why should a new "benchmarking suite" be added as opposed to simple use of existing workflow orchestration frameworks? It's very unclear what problem is trying to be solved. This appears to be one of the key points the authors wish to make, but it is both too brief and not clear.

Thank you, we have updated the manuscript to make the argument stronger about what we envision for 'benchmarking systems', and that benchmarking is certainly not simply running a workflow (although running a workflow is an important component). In particular, we have clarified how benchmarking systems go beyond workflow managers; and how benchmarking-system-powered benchmarks can be formalized. Changes can be found within the 'Introduction' and 'Why a benchmarking system' sections.

Third, the review of the landscape of benchmarks seems very biased towards sc-sequencing, and for example many benchmarks in other areas (e.g. GIAB, DREAM, etc.) appear to already cover many aspects of these analyses. As a simple example, DREAM studies have generally provided extensive "dataset- and/or metric-specific performance measures". For example, look at the cancer evolution DREAM this year in Nature Biotech from Boutros + Van Loo and the gene-reg DREAM this month in Nature Biotech from Meyer + Boer. It appears much of what is being called for appears to be standard practice in a large number of (high-profile, uncited!) benchmarking papers. Indeed on reviewing the former it even looks like they presented a simple decision-support tool for multi-task optimization. Overall, the authors claim there are "no standards" for benchmarking, which the proliferation of benchmarking organizations with their long track record of... standardized benchmarking shows!

We have updated the manuscript (section 'Benchmark stakeholders') to broaden benchmarking examples to include DREAM and GIAB, hence going beyond single-cell RNA-seq, and to emphasize the gap between benchmarking and benchmarking through benchmarking systems. We are aware of many high quality benchmarks that follow good practices; independently, we note that there is an academic trend for them to embrace benchmarking system-powered benchmarks, mainly motivated by reusability and continuous benchmarking. For instance, the Boutros paper is reporting on a study that started in 2014 and only has results up to 2021, and hence is far from indicative of recent algorithmic innovation. Recent papers switching to benchmarking-system powered benchmarks include:

- Systematic assessment of long-read RNA-seq methods for transcript identification and quantification (peer review: they switch to OpenEBench to address reviewer comment R.3.1) https://www.nature.com/articles/s41592-024-02298-3#Sec24
- Polyadenylation sites from the Zavolan group also uses OpenEBench: https://rnajournal.cshlp.org/content/29/12/1839.full
- Also Quest for Orthologs uses OpenEBench: https://academic.oup.com/nar/article/50/W1/W623/6584783

The DREAM paper also discusses (emphasis ours):

> ”To continually improve genomics models, **there is a need for standardized, robust benchmarking datasets**. The DREAM Challenge dataset addresses this need and the impact that such standardized datasets can have was demonstrated by the generalizability of DREAM-optimized models across different Drosophila and human datasets and tasks without additional model tuning. Nonetheless, **it should be noted that the models stemming from this challenge explored only a fraction of the possible design space and are likely to be improved upon**. Furthermore, performance of the DREAM-optimized models can be **optimized for different datasets by tailoring hyperparameters** of these models to the dataset in question or by using ensembles of the models.”

We are aware of the effort needed to automate some aspects of benchmarking (typically, the workflow orchestration); we believe benchmarking systems can facilitate these tasks (reducing human/time cost) while also facilitating reusability and ensuring transparency, hence building trust. We think decision-making tools borrowed from the operations science field (e.g. multicriteria decision making analysis) can be coupled to the process, hence being part of the automation. Benchmark conceptualization and interpretation cannot be automated, but can be eased and broadened: we believe benchmarking-system powered benchmarks can facilitate re-interpretation and meta-analysis. We have updated the manuscript to discuss these aspects (Figure 1 and content-wise), addressing also the multi-task optimization comment.

Overall, it's not clear to this reviewer what new space this article fills, and that it would make a substantive contribution to the education and discourse around benchmarking computational methods in biology and biomedicine.

Thank you, we have updated the manuscript (mainly Figure 1) to spell out the strengths and limitations of each component of the benchmarking process, from workflow orchestration to software environment control to result sharing, and their impact in workflow reproducibility, reusability, trust, and long-term extensibility.

Reviewer #2

Reviewer #2: Mallona and colleagues have produced a perspective manuscript on benchmarking in bioinformatics. I genuinely appreciate and like the content of this publication as it touches on various important aspects that people are often unaware of when performing a benchmarking exercise.

I have a couple of main comments, but overall, I'm happy with the manuscript as it is.

1) I'd reinforce the role of the communities in building unbiased (or at least less biased) benchmarking efforts. If we take the different "Critical Assessment" efforts, starting with CASP, we can see the value of the communities in sustaining and reflecting the existing challenge in a given field.

Thank you, we have updated the manuscript (section 'Building trust') to highlight the community part. We believe benchmarking systems can lower the barrier to contributing. If it is easy to add a new metric, it is more likely that someone will do it than if it takes a lot of effort, hence making the efforts less likely to be biased.

2) As we move towards more computationally intensive methods - especially with the emergence and adoption of AI technologies in Bioinformatics - it would also be essential to discuss the evaluation effort in terms of technical performance. Ideally, one would like to see the scientific performance of a given method together with its technical metrics, e.g. CPU/GPU usage, memory, storage, etc. This information could contribute to users making informed decisions on the best system for them.

We have updated the manuscript (Section 'Scope and interpretation of benchmarks') to extend the performance profiling aspects.

3) Recent efforts have translated the FAIR principles for research software, including scientific workflows, and how to measure them. This aspect is vital as it contributes to bioinformatics pipelines' long-term availability, use, and reproducibility. Nowadays, bioinformatics analyses result from a composition of different pieces of software.

We agree following the FAIR guidelines would reduce abandonware, as discussed in sections 'Benchmarks can be formally defined' and 'Open data formats and standards'.

4) It would be important to acknowledge that not all software can be readily available and distributed. Let's take the Quest for Orthologs initiative as an example. Many benchmarked systems are customised pipelines that are part of complex production systems and, therefore, are not shareable. I'm mentioning this aspect to reflect that benchmarking efforts have different levels of complexity and ways to be carried out.

We agree that long-term availability, (re)use and reproducibility can be facilitated by benchmarking systems by making the workflow part of the benchmarking process open sourced (free software) while also requiring individual steps (benchmarking modules) to be free software. So, FAIR principles can be followed both in the “frontend” (benchmark) and the backend (orchestration, sharing, etc). We have added Figure 1 to summarize the features, and consequences, of technical and ontological benchmarking choices, including whether the benchmarking system is to be used by a single user (solo benchmarking) or within a community.

We are also aware of somewhat divergent trends in software distribution paradigms, with a focus on environment handling packaging (e.g. conda) or containerization (e.g. docker) in bioinformatics; and on distributing software installations themselves on high performance computing (e.g. CernVMFS, computecanada). We have extended the section 'Reproducible software environments’ to elaborate, also in line with Reviewer #3’s comments.

Reviewer #3

Reviewer #3: The authors describe a concept for building a continuous benchmarking ecosystem in bioinformatics. The manuscript nicely summarizes generic formal steps of benchmarking and mentions the major challenges associated with making benchmarking continuous and sustainable.

There is not much to ask about in this manuscript. The text is pretty comprehensive, listing all the major challenges I could think of as well. Some comments can be found below.

Comments:

1. Page 5 bottom: unlike the last paragraph states, I would argue that the software environment is part of a workflow definition and not something in addition.

Thank you, we have updated the subsection 'Reproducible software environments’ and the section 'Benchmarks can be formally defined’ to be consistent. We believe that, from a benchmarking system perspective, a 'pipeline’ (sequence of tasks communicating via input-output file chains) is primarily defined by its topology - regardless of execution details. During execution, tasks are commands (calls) happening in (hopefully controlled) software environments. Modern workflow managers, e.g. Snakemake, make execution possible by elegantly sorting out the topology and the process calls while bundling them to software backends (e.g. conda, envmodules, singularity), even taking into account hardware requirements. Nonetheless, we think a benchmarking system should decouple the topology from the software backend to provide extra flexibility. So, for instance, a given benchmarking workflow (e.g. simulate -> preprocess -> run method with a parameter grid -> run metric) can be used to run a benchmark; but this same topology (in our terms, workflow) can be run with different software backends to benchmark the effect of the backend itself.

2. Page 6: I see how workflow management does not by default cover all aspects of benchmarking, namely ensuring appropriate hardware, managing access control, and versioning code and runs, as well as providing extensive documentation. However, I cannot follow the claim that there is no platform that uses a formalization language that covers all necessary aspects. Example: I can see how e.g. Snakemake does not cover e.g. versioning. But I would argue that something like that does not really need to be part of the formalization language (how would that happen?), but rather part of the platform in which the benchmark is conducted. For example, in ncbench, all of the mentioned aspects are easily covered by standard tools designed for the respective purpose (versioning with git, visualization with datavzrd, storage with zenodo, access control with zenodo and github, ...). How

would another formalization language help there? Maybe the authors could be more concrete to better explain what is missing there.

Thank you, some of these concerns are in line with Reviewer #1's. We have updated the manuscript and added Figure 1 to address workflow-specific, benchmark-specific and benchmarking-system-specific features, their scopes and formalization.

Regarding versioning, we have updated the manuscript to clarify the difference between benchmark versions and benchmark component versions in section 'There are various hidden design tradeoffs'.

As for hardware, and in line with Reviewer #2's comments, we have discussed how to characterize hardware homogeneity and specs when resource profiling. To us, this is a complex topic because ensuring hardware homogeneity requires some control in either provisioning or sending jobs to a queue with nodes fulfilling some requirements (i.e. having GPUs). Some benchmarks, particularly solo benchmarks, could be run locally / in whichever hardware is available to the benchmarker, without specific provisioning. Hence, we reason that the strategy to guarantee hardware comparability has to do with using hardware characterization suites (e.g. microarchitecture, GPU availability etc) and logging, instead of selecting the appropriate hardware before execution. So these hardware characteristics logs are then validated/evaluated/reported jointly to benchmarking results and performance metrics. We have updated the section 'Scope and interpretation of benchmarks' accordingly.

3. Page 10: in the list of software management automation methods, I miss conda as the most widely used approach nowadays. In particular in connection with systematically exploring version combinations, it has the advantage that it is much more lightweight and scalable than containers or also env-module related approaches like easybuild or cvmfs. With recent improvements there, (py-rattler/pixi) deploying conda packages is faster and more flexible than ever before. (sorry, this sounds like advertising, but I really think one should not simply skip it without mentioning it, and the improvements with rattler and pixi are actually massive). Moreover, conda offers an additional interesting aspect when thinking about the future of benchmarking: conda packages can in principle be compiled to web assembly (currently being explored in the big conda communities), allowing them to be executed within the browser. This could potentially also help with benchmarking, since it could allow benchmarks to be computed on-demand by the user without requiring any local deployment process nor cloud computing costs.

Thank you, we have updated the manuscript (Section 'Reproducible software environments') to discuss package managers, conda, containers and cernvmfs - also highlighting rattler, pixi and wasm for conda.

4. One of the biggest challenges with benchmarking is the burden of setting it up properly, and in consequence of maintaining it sustainably. This is the main reason why people tend to do all these abandonware software evaluations in their methods papers, and even benchmark studies are almost never designed to be repeated (one of the reasons why we had to start from scratch with ncbench). Maybe the manuscript should talk about automating and assisting with benchmark design, setup, development, and deployment. This could even discuss the potential of AI in this context, and the ability to leverage common knowledge and resources across different kinds of benchmarks that are related (e.g. between read mapping benchmarks and variant calling benchmarks).

We agree and are aware that one advantage of free software and modular benchmarks is reuse beyond the original use: once someone builds a module to, e.g., compute an adjusted Rand index score, it can be imported and reused by other benchmark(er)s. We believe LLMs could be used to help the creation of snippets to interface arbitrary input file shapes to these reused modules; and also to compose larger benchmarks from smaller ones (e.g. a benchmark for read alignment and quantification in RNA-seq could be prepended to a benchmark for differential gene expression starting on count tables). We have updated the manuscript accordingly in section 'Software licenses and attribution'.

Similarly, we are aware that explicit benchmark formalization can help meta-benchmarking, e.g. carrying out studies on the benchmarking processes, and/or to describe and analyze benchmarks (their layouts, aims, contribution patterns, biases etc) themselves. Our new Figure 1 incorporates the beyond-intended-use capabilities of benchmark-system-assisted benchmarks.