A meta-analysis of bioinformatics software benchmarks reveals major influences on software accuracy

Paul P. Gardner[1,2*], James M. Paterson[3], Stephanie McGimpsey[4], Fatemeh Ashari-Ghomi[5], Sinan U. Umu[6], Aleksandra Pawlik[7], Alex Gavryushkin[8], Michael A Black[1]

Abstract

Computational biology provides widely used and powerful software tools for testing and making inferences about biological data. In the face of rapidly increasing volumes of data, heuristic methods that trade software speed for accuracy may be employed. We are have studied these trade-offs using the results of a large number of independent software benchmarks, and evaluated whether external factors are indicative of accurate software.

We have extracted accuracy and speed ranks from independent benchmarks of different bioinformatic software tools, and evaluated whether the speed, author reputation, journal impact, recency and developer efforts are indicative of accuracy.

We found that software speed, author reputation, journal impact, number of citations and age are all unreliable predictors of software accuracy. This is unfortunate because citations, author and journal reputation are frequently cited reasons for selecting software tools. However, GitHub-derived records and high version numbers show that the accurate bioinformatic software tools are generally the product of many improvements over time, often from multiple developers.

We also find that the field of bioinformatics has a large excess of slow and inaccurate software tools, and this is consistent across many sub-disciplines. Meanwhile, there are few tools that are middle-of-road in terms of accuracy and speed trade-offs. We hypothesise that a form of publication-bias influences the publication and development of bioinformatic software. In other words, software that is intermediate in terms of both speed and accuracy may be difficult to publish - possibly due to author, editor and reviewer practices. This leaves an unfortunate hole in the literature as the ideal tools may fall into this gap. For example, high accuracy tools are not always useful if years of CPU time are required, while high speed is not useful if the results are also inaccurate.

[1] *Department of Biochemistry, University of Otago, Dunedin, New Zealand.*
[2] *Biomolecular Interaction Centre, University of Canterbury, Christchurch, New Zealand.*
[3] *Department of Civil and Natural Resources Engineering, University of Canterbury, Christchurch, New Zealand.*
[4] *Parasites and Microbes, Wellcome Sanger Institute, Wellcome Genome Campus, Hinxton, 8 Cambridgeshire, CB10 1RQ, UK.*
[5] *Research Group for Genomic Epidemiology, National Food Institute, Technical University of Denmark, Kongens Lyngby, Denmark.*
[6] *Department of Research, Cancer Registry of Norway, Oslo, Norway.*
[7] *New Zealand eScience Infrastructure, 49 Symonds St, Auckland, New Zealand.*
[8] *Department of Computer Science, University of Otago, Dunedin, New Zealand.*
*__Corresponding author__: paul.gardner@otago.ac.nz

Background

Computational biology software is widely used and has produced some of the most cited publications in the entire scientific corpus [1, 2, 3]. These highly-cited software tools include implementations of methods for sequence alignment and homology inference [4, 5, 6, 7], phylogenetic analysis [8, 9, 10, 11, 12], biomolecular structure analysis [13, 14, 15, 16, 17], and visualization and data collection [18, 19]. However, the popularity of a software tool does not necessarily mean that it is accurate or computationally efficient, instead usability, ease of installation, operating system support or other indirect factors may play a greater role in a software tool's popularity. Indeed, there have been several notable incidences where convenient, yet inaccurate software has caused considerable harm [20, 21, 22].

Progress in the biological sciences is increasingly limited by the ability to analyse large volumes of data, therefore the dependence of biologists on software is also increasing [23]. There is an increasing reliance on technological solutions for automating biological data generation (e.g. next-generation sequencing, mass-spectroscopy, cell-tracking and species tracking), therefore the biological sciences have become increasingly dependent upon software tools for processing large quantities of data [23]. As a consequence, the computational efficiency of data processing and analysis software is of great importance to decrease the energy, climate impact, and time costs of research [24]. Furthermore, as datasets become larger

even small error rates can have major impacts on the number of false inferences [25].

The gold-standard for determining accuracy is for researchers independent of individual tool development to conduct benchmarking studies, these benchmarks can serve a useful role in reducing the over-optimistic reporting of software accuracy [26, 27, 28] and the self-assessment trap [29, 30]. Benchmarking typically involves the use a number of positive and negative control datasets, so that predictions from different software tools can be partitioned into true or false groups, allowing a variety of metrics to be used to evaluate performance [31, 32, 28]. The aim of these benchmarks is to robustly identify tools that make acceptable compromises in terms of balancing speed with discriminating true and false predictions, and are therefore suited for wide adoption by the community.

For common computational biology tasks, a proliferation of software-based solutions often exists [33, 34, 35]. While this is a good problem to have, and points to a diversity of options from which practical solutions can be selected, having many possible options creates a dilemma for users. In the absence of any recent gold-standard benchmarks, how should scientific software be selected? In the following we presume that the "biological accuracy" of predictions is the most desirable feature for a software tool. Biological accuracy is the degree to which predictions or measurements reflect the biological truths based on expert-derived curated datasets (see Methods for the mathematical definition used here).

A number of possible predictors of software quality are used by the community of computational biology software users [36, 37, 38]. Some accessible, quantifiable and frequently used proxies for identifying high quality software include: **1. Recency:** recently published software tools may have built upon the results of past work, or be an update to an existing tool. Therefore these may be more accurate and faster. **2. Wide adoption:** a software tool may be widely used because it is fast and accurate, or because it is well-supported and user-friendly. In fact,"large user base", "word-of-mouth", "wide-adoption", "personal recommendation", and "recommendation from a close colleague", were frequent responses to surveys of "how do scientists select software?" [36, 37, 38]. **3. Journal impact:** high profile journals are run by editors and reviewers who carefully select and curate the best manuscripts. Therefore, high impact journals may be more likely to select manuscripts describing good software [39]. **4. Author/Group reputation:** the key to any project is the skills of the people involved, including maintaining a high collective intelligence [37, 40, 41]. As a consequence, an argument could be made that well respected and high-profile authors may write better software [42, 43]. **5. Speed:** software tools frequently trade accuracy for speed. For example, heuristic software such as the popular homology search tool, BLAST, compromises the mathematical guarantee of optimal solutions for more speed [4, 7]. Some researchers may naively interpret this fact as implying that slower software is likely to be more accurate. But speed may also be influenced by the programming language [44], and the level of hardware optimisation [45, 46]; However, the specific method of implementation generally has a greater impact (e.g., brute-force approaches versus rapid and sensitive pre-filtering [47, 48, 49]).

With the wide adoption of GitHub (47% of the 499 tools included in this study could be linked to a GitHub repository), and consequently quantifiable data on software development time and intensity indicators, such as the number of contributors to code, number of code changes and versions is available for these [50, 51, 52].

In the following study, we explore factors that may be indicative of software accuracy. This, in our opinion, should be one of the prime reasons for selecting a software tool. We have mined the large and freely accessible PubMed database [53] for benchmarks of computational biology software, and manually extracted accuracy and speed rankings for 499 unique software tools. For each tool, we have collected measures that may be predictive of accuracy, and may be subjectively employed by the research community as a proxy for software quality. These include relative speed, relative age, the productivity and impact of the corresponding authors, journal impact and the number of citations.

Results

We have collected relative accuracy and speed ranks for 499 distinct software tools. This software has been developed for solving a broad cross-section of computational biology tasks. Each software tool was benchmarked in at least one of 69 publications that satisfy the Boulesteix criteria [54]. In brief, the Boulesteix criteria are: 1. the main focus of the article is a benchmark. 2. the authors are reasonably neutral. 3. the test data and evaluation criteria are sensible.

For each of the publications describing these tools, we have (where possible) collected the journal's H5-index (Google Scholar Metrics), the maximum H-index and corresponding M-indices [42] for the corresponding authors for each tool, and the number of times the publication(s) associated with a tool has been cited using Google Scholar (data collected over a 6 month period in late 2020). Note that citation metrics are not static and will change over time. In addition, where possible we also extract the version number, the number of commits and number of contributors from GitHub repositories.

We have computed the Spearman's correlation coefficient for each pairwise combination of the mean normalised accuracy and speed ranks, with the year published, mean relative age (compared to software in the same benchmarks), journal H5 metrics, the total number of citations, the relative number of citations (compared to software in the same benchmarks) and the maximum H- and corresponding M-indices for the corresponding authors, version number, and numbers of commits and contributors. The results are presented in Figure 1A&B. We find significant associations between most of the citation-based metrics (journal H5, citations, relative citations, H-index and M-index). There is also a negative correlation between the

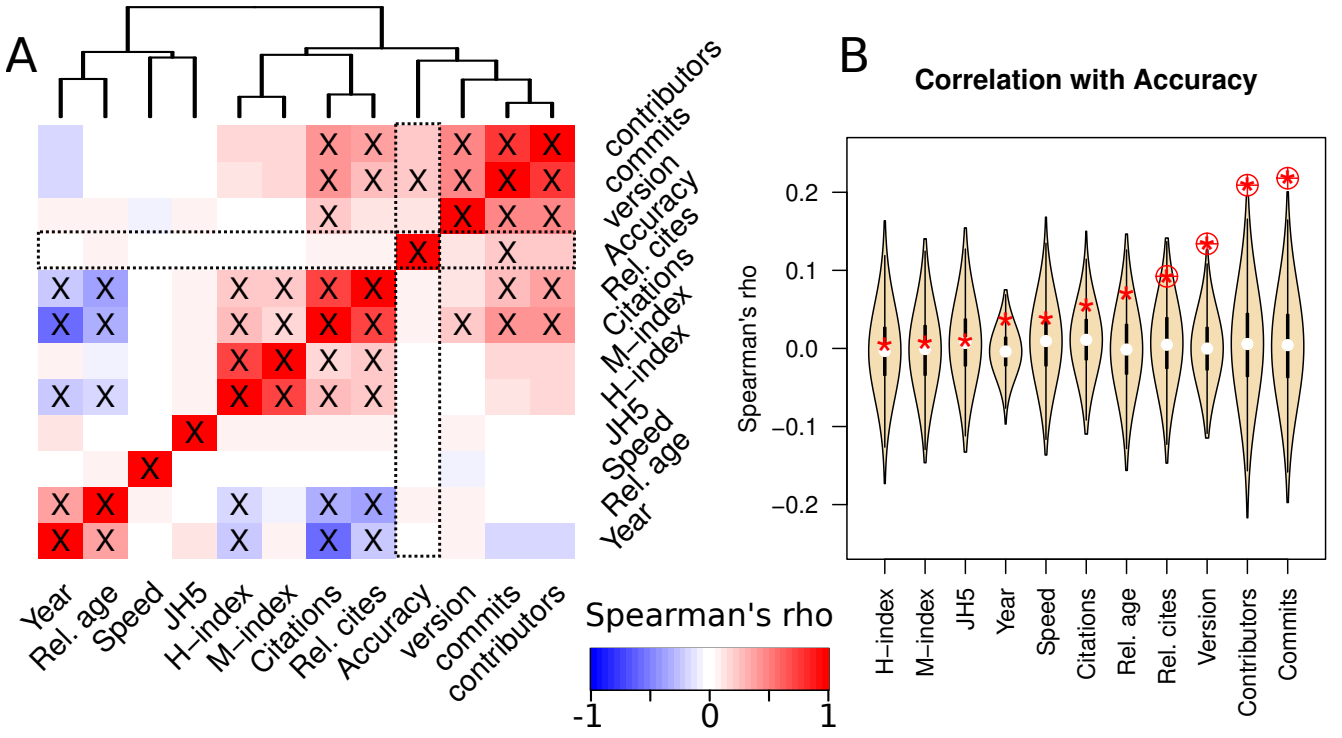

Figure 1. A. A heatmap indicating the relationships between different features of bioinformatic software tools. Spearman's rho is used to infer correlations between metrics such as citations based metrics, the year and relative age of publication, version number, GitHub derived activity measures, and the mean relative speed and accuracy rankings. Red colours indicate a positive correlation, blue colours indicate a negative correlation. Correlations with an P-value less than 0.05 (corrected for multiple-testing using the Benjami-Hochberg method) are indicated with a 'X' symbol. The correlations with accuracy are illustrated in more detail in **B**, the relationship between speed and accuracy is shown in more detail in **Figure 2**. **B.** Violin plots of Spearman's correlations between permuted accuracy ranks and different software tool features. The unpermuted correlations are indicated with a red asterix. For each benchmark, 1,000 permuted sets of accuracy and speed ranks were generated, and the ranks were normalised to lie between 0 and 1 (see Methods for details). Circled asterixs are significant (empirical P-value < 0.05, corrected for multiple-testing using the Benjami-Hochberg method).

year of publication, the relative age and many of the citation-based metrics.

Data on the number of updates to software tools from GitHub such as the number of versions, commits and contributors was significantly correlated with software accuracy (respective Spearman's rhos = 0.15, 0.22, 0.21, and respective Benjamini & Hochberg corrected P-values = 0.029, 0.042, 0.063), Figure 1B. The significance if these features was further confirmed with a permutation test (Figure 1B). These features were not correlated with speed however (see Figure 1A & Supplementary Figure ????). We also found that reputation metrics such as citations, author and journal H-indices, and the age of tools were generally **not** correlated with either tool accuracy or speed (Figure 1A&B).

In order to gain a deeper understanding of the distribution of available bioinformatic software tools on a speed versus accuracy landscape, we ran a permutation test. The ranks extracted from each benchmark were randomly permuted, generating 1,000 randomized speed and accuracy ranks. In the cells of a 3×3 grid spanning the normalised speed and accuracy ranks we computed a Z-score for the observed number of tools in a cell, compared to the expected distributions generated by 1,000 randomized ranks. The results of this are shown in Figure 2. We identified 4 of 9 bins where there was a significant excess or dearth of tools. For example, there was an excess of "slow and inaccurate" software (Z=3.39, P-value=3.5×10^{-4}), with more moderate excess of "slow and accurate" and "fast and accurate" software (Z=2.49 & 1.7, P=6.3×10^{-3} & 0.04 respectively). We find that only the "fast and inaccurate" extreme class is at approximately the expected proportions based upon the permutation test (Figure 2B).

The largest difference between the observed and expected software ranks is the reduction in the number of software tools that are classed as intermediate in terms of both speed and accuracy based on permutation tests (see Methods for details, Figure 2). The middle cell of Figure 2A and left-most violin plot of Figure 2B highlight this extreme, (Z=-6.38, P-value=9.0×10^{-11}).

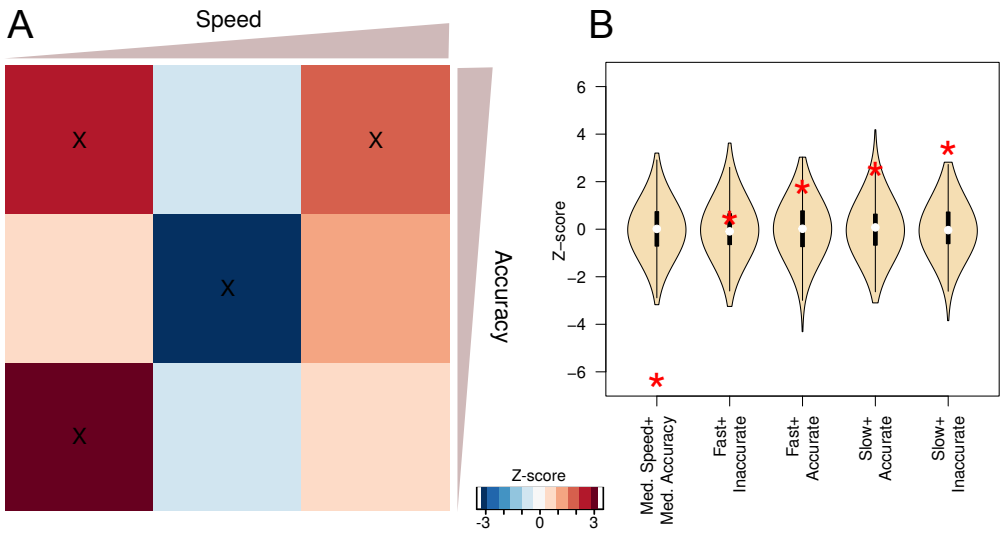

Figure 2. A. A heatmap indicating the relative paucity or abundance of software in the range of possible accuracy and speed rankings. Redder colours indicate an abundance of software tools in an accuracy and speed category, while bluer colours indicate scarcity of software in an accuracy and speed category. The abundance is quantified using a Z-score computation for each bin, this is derived from 1,000 random permutations of speed and accuracy ranks from each benchmark. Mean normalised ranks of accuracy and speed have been binned into 9 classes (a 3×3 grid) that range from comparatively slow and inaccurate to comparatively fast and accurate. Z-scores with a P-value less than 0.05 are indicated with a '$\mathbb{X}$'. **B.** The z-score distributions from the permutation tests (indicated with the wheat coloured violin plots) compared to the z-score for the observed values for each of the corner and middle square of the heatmap.

Conclusion

We have gathered data on the relative speeds and accuracies of 499 bioinformatic tools from 69 benchmarks published between 2005 and 2020. Our results provide significant support for the suggestion that there are major benefits to the long-term support of software development [55]. The finding of a strong relationship between the number of commits and code contributors to GitHub (i.e. software updates) and accuracy, highlights the benefits of long-term or at least intensive development.

Our study finds little evidence to support that impact-based metrics have any relationship with software quality, which is unfortunate, as these are frequently cited reasons for selecting software tools [38]. This implies that high citation rates for bioinformatic software [1, 2, 3] is more a reflection of other factors such as user-friendliness or the Matthew Effect [56, 57] other than accuracy.

We found the lack of a correlation between software speed and accuracy surprising. The slower software tools are over-represented at both high and low levels of accuracy (Figure 2). In addition, there is an large under-representation of software that has intermediate levels of both accuracy and speed. A possible explanation for this is that bioinformatic software tools are bound by a form of publication-bias [58, 59]. That is, the probability that a study being published is influenced by the results it contains [60]. The community of developers, reviewers and editors may be unwilling to publish software that is not highly ranked on speed or accuracy. If correct, this may have unfortunate consequences as these tools may never-the-less have further uses.

While we have taken pains to mitigate many issues with our analysis, nevertheless some limitations remain. For example, it has proven difficult to verify if the gap in medium accuracy and medium speed software is genuinely the result of publication bias, or due to additional factors that we have not taken in to account. In addition, all of the features we have used here are moving targets. For example, as software tools are refined, their relative accuracies and speeds will change, the citation metrics, ages, and version control derived measures also change over time. Here we report a snapshot of values from 2020. The benchmarks themselves may also introduce biases into the study. For example, there are issues with a potential lack of independence between benchmarks (e.g., shared datasets, metrics and tools), there are heterogeneous measures of accuracy and speed and often unclear processes for including different tools.

We propose that the full spectrum of software tool accuracies and speeds serves a useful purpose to the research community. Like negative results, if honestly reported this information, illustrates to the research community that certain approaches are not practical research avenues [61]. The current practices of publishers, editors, reviewers and authors of software tools therefore may be depriving our community of tools for building effective and productive workflows.

The most reliable way to identify accurate software tools is through neutral software benchmarks [54]. We are hopeful

that this, along with steps to reduce the publication-bias we have described, will reduce the over-optimistic and misleading reporting of tool accuracy [26, 27, 29].

Methods

In order to evaluate predictors of computational biology software accuracy, we mined the published literature, extracted data from articles, connected these with bibliometric databases, and tested for correlates with accuracy. We outline these steps in further detail below.

Criteria for inclusion: We are interested in using computational biology benchmarks that satisfy Boulesteix's (ALB) three criteria for a "neutral comparison study" [54]. Firstly, the main focus of the article is the comparison and **not** the introduction of a new tool. Secondly, the authors should be reasonably neutral, which means that the authors should not generally have been involved in the development of the tools included in the benchmark. Thirdly, the test data and evaluation criteria should be sensible. This means that the test data should be independent of data that tools have been trained upon, and that the evaluation measures appropriately quantify correct and incorrect predictions.

Literature mining: We identified an initial list of 10 benchmark articles that satisfy the ALB-criteria. These were identified based upon previous knowledge of published articles and were supplemented with several literature searches (e.g., ["benchmark" AND "cputime"] was used to query both GoogleScholar and Pubmed [53, 62]). We used these articles to seed a machine-learning approach for identifying further candidate articles and to identify new search terms to include.

For our machine-learning-based literature screening, we computed a score, $s(a)$, for each article that tells us the likelihood that it is a benchmark. In brief, our approaches uses 3 stages:

1. Remove high frequency words from the title and abstract of candidate articles (e.g. 'the', 'and', 'of', 'to', 'a', ...)
2. Compute a log-odds score for the remaining words
3. Use a sum of log-odds scores to give a total score for candidate articles

For stage 1, we identified a list of high frequency (e.g. $f(\text{word}) > 1/10,000$) words by pooling the content of two control texts [63, 64].

For stage 2, in order to compute a log-odds score for bioinformatic words, we computed the frequency of words that were not removed by our high frequency filter in two different groups of articles: bioinformatics-background and bioinformatics-benchmark articles. The text from bioinformatics-background articles were drawn from the bioinformatics literature, but these were not necessarily associated with benchmark studies. For background text we used Pubmed ([53, 62] to select 8,908 articles that contained the word "bioinformatics" in the title or abstract and were published between 2013 and 2015. We computed frequencies for each word by combining text from titles and abstracts for the background and training articles. A log-odds score was computed for each word using the following formula:

$$lo(word) = \log_2 \frac{f_{tr}(word) + \delta}{f_{bg}(word) + \delta}$$

Where δ was a pseudo-count added for each word ($\delta = 10^{-5}$, by default), $f_{bg}(word)$ and $f_{tr}(word)$ were the frequencies of a *word* in the background and training datasets respectively. Word frequencies were computed by counting the number of times a word appears in the pool of titles and abstracts, the counts were normalised by the total number of words in each set.

Thirdly, we also collected a group of candidate benchmark articles by mining Pubmed for articles that were likely to be benchmarks of bioinformatic software, these match the terms: "((bioinformatics) AND (algorithms OR programs OR software)) AND (accuracy OR assessment OR benchmark OR comparison OR performance) AND (speed OR time)". Further terms used in this search were progressively added as relevant enriched terms were identified in later iterations. The final query is given in **supplementary materials**.

A score is computed for each candidate article by summing the log-odds scores for the words in title and abstract, i.e. $s(a) = \sum_i^N lo(w_i)$. The high scoring candidate articles are then manually evaluated against the ALB-criteria. Accuracy and speed ranks were extracted from the articles that met the criteria, and these were added to the set of training articles. The evaluated candidate articles that did not meet the ALB-criteria were incorporated into the set of background articles. This process was iterated and resulted in the identification of 69 benchmark articles, containing 134 different benchmarks. Together these ranked 499 distinct software packages.

There is a potential for bias to have been introduced into this dataset. Some possible forms of bias include converging on a niche group of benchmark studies due to the literature mining technique that we have used. A further possibility is that benchmark studies themselves are biased, either including very high performing or very low performing software tools. To address each of these concerns we have attempted to be as comprehensive as possible in terms of benchmark inclusion, as well as including comprehensive benchmarks (i.e., studies that include all available software tools that address a specific biological problem).

Data extraction and processing: for each article that met the ALB-criteria and contained data on both the accuracy and speed from their tests, we extracted ranks for each tool. Many articles contained multiple benchmarks, in these cases we selected a range of these, the provenance of which is stored with the accuracy metric and raw speed and accuracy rank data for each tool. In line with rank-based statistics, the cases where tools were tied were resolved by using a midpoint rank (e.g., if tool 3 and 4 are tied, the rank 3.5 was used) [65]. Each

rank extraction was independently verified by at least one other co-author to ensure both the provenance of the data could be established and that the ranks were correct. The ranks for each benchmark were then normalised to lie between 0 and 1 using the formula $1 - \frac{r-1}{n-1}$ where 'r' is a tool's rank and 'n' is the number of tools in the benchmark. For tools that were benchmarked multiple times with multiple metrics (e.g., BWA was evaluated in 6 different articles [66, 67, 68, 69, 70, 71]) a mean normalised rank was used to summarise the accuracy and speed performance. Or, more formally:

$$accuracy = \sum_{i=1..N} 1 - \frac{r_i^{accuracy} - 1}{n_i - 1},$$

$$speed = \sum_{i=1..N} 1 - \frac{r_i^{speed} - 1}{n_i - 1}$$

For each tool we identified the corresponding publications in GoogleScholar; the total number of citations was recorded, the corresponding authors were also identified, and if they had public GoogleScholar profiles, we extracted their H-index and calculated a M-index ($\frac{H-index}{y}$) where 'y' is the number of years since their first publication. The journal quality was estimated using the H5-index from GoogleScholar Metrics.

The year of publication was also recorded for each tool. "Relative age" and "relative citations" were also computed for each tool. For each benchmark, software was ranked by year of first publication (or number of citations), ranks were assigned and then normalised as described above. Tools ranked in multiple evaluations were then assigned a mean value for "relative age" and "relative citations".

The papers describing tools were checked for information on version numbers and links to GitHub. Google was also employed to identify GitHub repositories. When a repository was matched with a tool, the number of "commits" and number of "contributors" was collected, when details of version numbers were provided, these were also harvested. Version numbers are inconsistently used between groups, and may begin at either 0 or 1. To counter this issue we have added '1' to all versions less than '1', for example, version 0.31 become 1.31. In addition, multiple point releases may be used e.g. 'version 5.2.6', these have been mapped to the nearest decimal value '5.26'.

Statistical analysis: For each tool we manually collected up to 12 different statistics from GoogleScholar, GitHub and directly from literature describing tools (1. corresponding author's H-index, 2. corresponding author's M-index, 3. journal H5 index, 4. normalised accuracy rank, 5. normalised speed rank, 6. number of citations, 7. relative age, 8. relative number of citations, 9. year first published, 10. version 11. number of commits to GitHub, 12. number of contributors to GitHub). These were evaluated in a pairwise fashion to produce Figure 1 A&B, the R code used to generate these is given in a GitHub repository (linked below).

For each benchmark of three or more tools, we extracted the published accuracy and speed ranks. In order to identify whether there was an enrichment of certain accuracy and speed pairings we constructed a permutation test. The individual accuracy and speed ranks were reassigned to tools in a random fashion and each new accuracy and speed rank pairing was recorded. For each benchmark this procedure was repeated 1,000 times. These permuted rankings were normalised and compared to the real rankings to produce the '╳' points in Figure 1B and the heatmap and histograms in Figure 2. The heatmap in Figure 2 is based upon Z-scores ($Z = \frac{x-\bar{x}}{s}$). For each cell in a 3×3 grid a Z-score (and corresponding P-value is computed, either with the 'pnorm' distribution function in R (Figure 2A) or empirically (Figure 2B)) is computed to illustrate the abundance or lack of tools in a cell relative to the permuted data.

Data availability

Raw datasets, software and documents are available under a CC-BY license:
`https://github.com/Gardner-BinfLab/speed-vs-accuracy-meta-analysis`

Acknowledgements

The authors acknowledge the valued contribution of invaluable discussions with Anne-Laure Boulesteix, Shinichi Nakagawa, Suetonia Palmer and Jason Tylianakis. Murray Cox, Raquel Norel, Alexandros Stamatakis, Jens Stoye,Tandy Warnow, provided valuable feedback on drafts of the manuscript.

PPG is supported by a Rutherford Discovery Fellowship, administered by the Royal Society Te Apārangi, PPG, AG and MB acknowledge support from a Data Science Programmes grant (UOAX1932).

This work was largely conducted on the traditional territory of Kāi Tahu.

References

[1] Carolina Perez-Iratxeta, Miguel A Andrade-Navarro, and Jonathan D Wren. Evolving research trends in bioinformatics. *Brief. Bioinform.*, 8(2):88–95, March 2007.

[2] Richard Van Noorden, Brendan Maher, and Regina Nuzzo. The top 100 papers. *Nature*, 514(7524):550–553, 30 October 2014.

[3] Jonathan D Wren. Bioinformatics programs are 31-fold over-represented among the highest impact scientific papers of the past two decades. *Bioinformatics*, 5 May 2016.

[4] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, October 1990.

[5] J D Thompson, D G Higgins, and T J Gibson. CLUSTAL w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22(22):4673–4680, November 1994.

[6] J D Thompson, T J Gibson, F Plewniak, F Jeanmougin, and D G Higgins. The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res.*, 25(24):4876–4882, 15 December 1997.

[7] S F Altschul, T L Madden, A A Schäffer, J Zhang, Z Zhang, W Miller, and D J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–3402, 1 September 1997.

[8] Joseph Felsenstein. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39(4):783–791, 1 July 1985.

[9] N Saitou and M Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4(4):406–425, July 1987.

[10] D Posada and K A Crandall. MODELTEST: testing the model of DNA substitution. *Bioinformatics*, 14(9):817–818, 1998.

[11] Fredrik Ronquist and John P Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574, 12 August 2003.

[12] Koichiro Tamura, Joel Dudley, Masatoshi Nei, and Sudhir Kumar. MEGA4: Molecular evolutionary genetics analysis (MEGA) software version 4.0. *Mol. Biol. Evol.*, 24(8):1596–1599, August 2007.

[13] G M Sheldrick. Phase annealing in SHELX-90: direct methods for larger structures. *Acta Crystallogr. A*, 46(6):467–473, 1 June 1990.

[14] George M Sheldrick. A short history of SHELX. *Acta Crystallogr. A*, 64(Pt 1):112–122, January 2008.

[15] T A Jones, J Y Zou, S W Cowan, and M Kjeldgaard. Improved methods for building protein models in electron density maps and the location of errors in these models. *Acta Crystallogr. A*, 47 (Pt 2):110–119, 1 March 1991.

[16] R A Laskowski, M W MacArthur, D S Moss, and J M Thornton. PROCHECK: a program to check the stereochemical quality of protein structures. *J. Appl. Crystallogr.*, 26(2):283–291, 1 April 1993.

[17] Zbyszek Otwinowski and Wladek Minor. [20] processing of x-ray diffraction data collected in oscillation mode. In *Methods in Enzymology*, volume Volume 276, pages 307–326. Academic Press, 1997.

[18] P J Kraulis. MOLSCRIPT: a program to produce both detailed and schematic plots of protein structures. *J. Appl. Crystallogr.*, 24(5):946–950, 1 October 1991.

[19] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, T N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic Acids Res.*, 28(1):235–242, 1 January 2000.

[20] Nancy G Leveson and Clark S Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.

[21] ML Cummings and David Britton. Regulating safety-critical autonomous systems: past, present, and future perspectives. In *Living with robots*, pages 119–140. Elsevier, 2020.

[22] J Herkert, J Borenstein, and K Miller. The boeing 737 max: Lessons for engineering ethics. *Sci Eng Ethics*, 26(6):2957–2974, Dec 2020.

[23] Vivien Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–260, 13 June 2013.

[24] Joel Gombiner. Carbon footprinting the internet. *Consilience-The Journal of Sustainable Development*, 5(1), 2011.

[25] John D Storey and Robert Tibshirani. Statistical significance for genomewide studies. *Proc. Natl. Acad. Sci. U. S. A.*, 100(16):9440–9445, 5 August 2003.

[26] Anne-Laure Boulesteix. Over-optimism in bioinformatics research. *Bioinformatics*, 26(3):437–439, 1 February 2010.

[27] Monika Jelizarow, Vincent Guillemot, Arthur Tenenhaus, Korbinian Strimmer, and Anne-Laure Boulesteix. Over-optimism in bioinformatics: an illustration. *Bioinformatics*, 26(16):1990–1998, 15 August 2010.

[28] L M Weber, W Saelens, R Cannoodt, C Soneson, A Hapfelmeier, P P Gardner, A L Boulesteix, Y Saeys, and M D Robinson. Essential guidelines for computational method benchmarking. *Genome Biol*, 20(1):125, 06 2019.

[29] Raquel Norel, John Jeremy Rice, and Gustavo Stolovitzky. The self-assessment trap: can we all be better than average? *Mol. Syst. Biol.*, 7(1):537, 1 January 2011.

[30] Stefan Buchka, Alexander Hapfelmeier, Paul P Gardner, Rory Wilson, and Anne-Laure Boulesteix. On the optimistic performance evaluation of newly introduced bioinformatic methods. *MetaArXiv*, Jan 2021.

[31] J P Egan. *Signal Detection Theory and ROC-analysis*. Series in Cognition and Perception. Academic Press, New York, 1975.

[32] T Hall, S Beecham, D Bowes, D Gray, and S Counsell. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans. Software Eng.*, 38(6):1276–1304, November 2012.

[33] Joseph Felsenstein. Phylogeny programs. *Internet address: http://evolution. gs. washington. edu/phylip/software. html*, 1995.

[34] Stephen Altschul, Barry Demchak, Richard Durbin, Robert Gentleman, Martin Krzywinski, Heng Li, Anton Nekrutenko, James Robinson, Wayne Rasband, James Taylor, and Cole Trapnell. The anatomy of successful computational biology software. *Nat. Biotechnol.*, 31(10):894–897, October 2013.

[35] Vincent J Henry, Anita E Bandrowski, Anne-Sophie Pepin, Bruno J Gonzalez, and Arnaud Desfeux. OMICtools: an informative directory for multi-omic data analysis. *Database*, 2014, 14 July 2014.

[36] Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, SECSE '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[37] Lucas N Joppa, Greg McInerny, Richard Harper, Lara Salido, Kenji Takeda, Kenton O'Hara, David Gavaghan, and Stephen Emmott. Troubling trends in scientific software use. *Science*, 340(6134):814–815, 17 May 2013.

[38] Nicholas Loman and Thomas Connor. Bioinformatics infrastructure and training survey, 2015.

[39] E Garfield. Citation indexes for science; a new dimension in documentation through association of ideas. *Science*, 122(3159):108–111, 15 July 1955.

[40] Anita Williams Woolley, Christopher F Chabris, Alex Pentland, Nada Hashmi, and Thomas W Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 29 October 2010.

[41] Kendra S Cheruvelil, Patricia A Soranno, Kathleen C Weathers, Paul C Hanson, Simon J Goring, Christopher T Filstrup, and Emily K Read. Creating and maintaining high-performing collaborative research teams: the importance of diversity and interpersonal skills. *Front. Ecol. Environ.*, 12(1):31–38, 1 February 2014.

[42] J E Hirsch. An index to quantify an individual's scientific research output. *Proc. Natl. Acad. Sci. U. S. A.*, 102(46):16569–16572, 15 November 2005.

[43] Lutz Bornmann, Rüdiger Mutz, and Hans-Dieter Daniel. Are there better indices for evaluation purposes than the h-index? a comparison of nine different variants of the h-index using data from biomedicine. *J. Am. Soc. Inf. Sci.*, 59(5):830–837, 1 March 2008.

[44] Mathieu Fourment and Michael R Gillings. A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics*, 9:82, 5 February 2008.

[45] Michael Farrar. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23(2):156–161, 15 January 2007.

[46] Lorenzo Dematté and Davide Prandi. GPU computing for systems biology. *Brief. Bioinform.*, 11(3):323–333, May 2010.

[47] J Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(11):1203–1212, November 1989.

[48] Christos H Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., Chichester, UK, 2003.

[49] Charles E Leiserson, Neil C Thompson, Joel S Emer, Bradley C Kuszmaul, Butler W Lampson, Daniel Sanchez, and Tao B Schardl. There's plenty of room at the top: What will drive computer performance after moore's law? *Science*, 368(6495), 2020.

[50] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 155–165, 2014.

[51] M G Dozmorov. Github statistics as a measure of the impact of open-source bioinformatics software. *Front Bioeng Biotechnol*, 6:198, 2018.

[52] Serghei Mangul, Thiago Mosqueiro, Dat Duong, Keith Mitchell, Varuni Sarwal, Brian Hill, Jaqueline Brito, Russell Jared Littman, Benjamin Statz, Angela Ka-Mei Lam, et al. A comprehensive analysis of the usability and archival stability of omics computational tools and resources. *bioRxiv*, page 452532, 2018.

[53] Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael Dicuccio, Scott Federhen, Michael Feolo, Lewis Y Geer, Wolfgang Helmberg, Yuri Kapustin, David Landsman, David J Lipman, Zhiyong Lu, Thomas L Madden, Tom Madej, Donna R Maglott, Aron Marchler-Bauer, Vadim Miller, Ilene Mizrachi, James Ostell, Anna Panchenko, Kim D Pruitt, Gregory D Schuler, Edwin Sequeira, Stephen T Sherry, Martin Shumway, Karl Sirotkin, Douglas Slotta, Alexandre Souvorov, Grigory Starchenko, Tatiana A Tatusova, Lukas Wagner, Yanli Wang, W John Wilbur, Eugene Yaschenko, and Jian Ye. Database resources of the national center for biotechnology information. *Nucleic Acids Res.*, 38(Database issue):D5–16, January 2010.

[54] Anne-Laure Boulesteix, Sabine Lauer, and Manuel J A Eugster. A plea for neutral comparison studies in computational sciences. *PLoS One*, 8(4):e61562, 24 April 2013.

[55] Adam Siepel. Challenges in funding and developing genomic software: roots and remedies. *Genome biology*, 20(1):1–14, 2019.

[56] Vincent Larivière and Yves Gingras. The impact factor's matthew effect: A natural experiment in bibliometrics. *J. Am. Soc. Inf. Sci.*, 61(2):424–427, 1 February 2010.

[57] Robert K Merton and Others. The matthew effect in science. *Science*, 159(3810):56–63, 1968.

[58] Anne-Laure Boulesteix, Veronika Stierle, and Alexander Hapfelmeier. Publication bias in methodological com-

putational research. *Cancer Inform.*, 14(Suppl 5):11–19, 15 October 2015.

[59] Silas Boye Nissen, Tali Magidson, Kevin Gross, and Carl T Bergstrom. Publication bias and the canonization of false facts. *Elife*, 5:e21451, 2016.

[60] Theodore D Sterling, Wilf L Rosenbaum, and James J Weinkam. Publication decisions revisited: The effect of the outcome of statistical tests on the decision to publish and vice versa. *The American Statistician*, 49(1):108–112, 1995.

[61] Daniele Fanelli. Negative results are disappearing from most disciplines and countries. *Scientometrics*, 90(3):891–904, 2012.

[62] J McEntyre and D Lipman. PubMed: bridging the information gap. *CMAJ*, 164(9):1317–1319, 1 May 2001.

[63] Lewis Carroll. *Alice's adventures in Wonderland*. Macmillan and Co., London, 1865.

[64] J R R Tolkien. *The Hobbit, Or, There and Back Again*. George Allen & Unwin, UK, 1937.

[65] H B Mann and D R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.*, 18(1):50–60, 1947.

[66] Suying Bao, Rui Jiang, Wingkeung Kwan, Binbin Wang, Xu Ma, and You-Qiang Song. Evaluation of next-generation sequencing software in mapping and assembly. *J. Hum. Genet.*, 56(6):406–414, June 2011.

[67] Ségolène Caboche, Christophe Audebert, Yves Lemoine, and David Hot. Comparison of mapping algorithms used in high-throughput sequencing: application to ion torrent data. *BMC Genomics*, 15:264, 5 April 2014.

[68] Ayat Hatem, Doruk Bozdağ, Amanda E Toland, and Ümit V Çatalyürek. Benchmarking short sequence mapping tools. *BMC Bioinformatics*, 14:184, 7 June 2013.

[69] Sophie Schbath, Véronique Martin, Matthias Zytnicki, Julien Fayolle, Valentin Loux, and Jean-François Gibrat. Mapping reads on a genomic sequence: an algorithmic overview and a practical comparative analysis. *J. Comput. Biol.*, 19(6):796–813, June 2012.

[70] Matthew Ruffalo, Thomas LaFramboise, and Mehmet Koyutürk. Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics*, 27(20):2790–2796, 15 October 2011.

[71] Manuel Holtgrewe, Anne-Katrin Emde, David Weese, and Knut Reinert. A novel and well-defined benchmarking method for second generation read mapping. *BMC Bioinformatics*, 12:210, 26 May 2011.