A meta-analysis of computational biology benchmarks reveals that publication bias that unduly influences speed and accuracy

Paul Gardner[1,2,3,*], James Paterson[1,2], Fatemeh Ashari Ghomi[1,2], Sinan Uğur Umu[1,2], Stephanie McGimpsey[4,5]

[1]School of Biological Sciences, [2]Biomolecular Interaction Centre, [3]Bio-Protection Research Centre, University of Canterbury, Christchurch, New Zealand

[4]McConomy School of Dance, Derry, Ireland

[5]Research for Good, Limavady, Ireland

*paul.gardner@canterbury.ac.nz

Abstract

Computational biology has provided widely used and powerful tools for testing and making inferences about biological data. However, how reliable are these tools? Do they provide good trade-offs between speed and accuracy? What factors are predictive of good software?
In this work we mine published benchmarks of computational biology tools, we collect data on the relative accuracy and speed of these tools and then test to see what factors influence accuracy e.g. speed, author reputation, journal impact or the number of citations.
We found that none of these measures are reliable predictors of software accuracy. We did find that there is an excess of slow and inaccurate software tools, while there are very few tools of middling accuracy and speed. We hypothesise that this is due to a publication bias i.e. at present it is difficult to publish methods that are neither the fastest or the most accurate.

Computational biology software is very widely used and has produced some of the most cited publications in the scientific corpus (1–3). This software includes methods for sequence alignment and homology inference (4–7), phylogenetic analysis (8–12), statistical analysis of survival patterns in biomedicine (13, 14), biomolecular structure analysis (15–19), visualization and data collection (20, 21). Yet the popularity of computational tools or software suites does not necessarily imply these methods are accurate or computationally efficient.

There is an increasing use of engineering and technology solutions for automating biological data generation (e.g. NGS, qPCR, MS, cell-tracking, site monitoring & species tracking), therefore the biological sciences have become increasingly dependent upon computational methods for processing large quantities of data (22). As a consequence computational efficiency of analysis tools is of increasing importance to decrease energy and time costs (23), similarly even small error rates can have a major impact

The gold-standard for determining accuracy is for independent researchers to conduct benchmarks, which can serve a useful role in reducing the over-optimistic reporting of software accuracy (24, 25) and the self-assessment trap (26). Benchmark studies typically use a number of positive and negative control datasets, predictions can then be partitioned into true or false groups and a variety of metrics can be used to evaluate the performance of different predictions (27, 28). Some benchmarks now use live, or frequently updated, results to illustrate the latest developments in software performance e.g. (29–31). The aim of this research is to independently identify tools that make acceptable compromises in terms of false predictions, true predictions and speed, and are therefore suited for wide adoption by the community.

For common computational biology tasks, a proliferation of software-based solutions often exists (32–36). While this may generally be a good problem to have, and points to a diversity of options from which practical solutions can be selected, 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 work, we presume that biological accuracy is the most desirable feature of software.

A number of possible predictors of software quality are used by the community of computational biology software users. Some accessible, quantifiable and frequently used proxies for identifying high quality software include: **1. Recency:** a recently published method is likely to have built upon the results of past methods. In principle, these may therefore be more accurate and/or faster. **2. Wide adoption:** a method may be widely used because it is fast and accurate or alternatively, because it's very user-friendly. The related measures, word-of-mouth and wide-adoption were frequent responses to "how do scientists select software?" surveys (37). **3. Journal impact:** high profile journals are run by editors and reviewers who devote much effort to curating their publications. Therefore, high impact journals may be more likely to select manuscripts describing good methods, alternatively good methods may be more likely to be submitted to good journals (38). **4. Author/Group reputation:** the key to any project is the skills of the people involved, including maintaining a high collective intelligence (39, 40). As a consequence, an argument could be made that well respected and high-profile authors will produce better software (41, 42). **5. Speed:** software is frequently said to trade accuracy for speed. For example, heuristic methods such as the popular homology search tool, BLAST, compromise the mathematical guarantee of optimal solutions for more speed (4, 7). Some researchers may naively interpret this fact as slower likely to be more accurate. Speed is influenced by the programming language (43), however the implementation is likely to have more of an impact (e.g. brute-force approaches versus rapid and sensitive pre-filtering (44, 45)).

Other factors that influence whether a software tool is selected include: whether the documentation is good, user-friendly, word-of-mouth and "used in a similar analysis" (37), this sort of information is not as readily quantifiable as the above measures. However, citation metrics may be a useful proxy. The word-of-mouth factor may also explain the reason why some software continues to be used, in spite of poor relative performance e.g. (46).

In the following study, we have investigated predictors of algorithm accuracy. This, in our opinion, should be one of the prime reasons for selecting a software tool. We have mined the PubMed database (47) for benchmarks of computational biology software, and manually extracted accuracy and speed rankings for each more than 240 methods. For each method we have collected measures that may be predictive of accuracy, and may be employed by the researcher community as a proxy for software quality. These include relative speed, relative age, the productivity and impact of corresponding authors, journal impact and the number of citations.

Results

We have collected relative accuracy and speed ranks for 243 distinct software methods. These methods have been developed for solving a broad cross-section computational biology tasks. These include methods for homology search (48), genome sequence analysis (e.g. read mapping or sequence assembly) (49–62), multiple sequence alignment (63–67), cell tracking (68), transcriptome analysis (69–72), RNA interaction prediction (73), protein interactions (74), protein structure prediction (75, 76), epistasis (77), metagenomic analysis (78, 79), repetitive sequence prediction (80), proteomics (81, 82) and phylogenetics (83–87). Each method was benchmarked in at least one of 43 benchmarks that satisfy the Boulesteix criteria (88).

For each of the publications describing these methods we have (when possible) identified the 2014 journal impact factor, published by Thomson Reuters (89) and the H5-index published by Google Scholar Metrics. We have collected the H-indices and M-indices (41) for the corresponding authors for each method, and the number of times the publication(s) associated with a method has been cited using Google Scholar (data collected over a 1 month period in early 2016).
We have computed the Spearman's correlation coefficient (ρ) for each pairwise combination of the mean normalised accuracy and speed ranks, the year published, mean relative age (compared to tools in the same benchmarks), journal IF and H5 metrics, the total number of citations, the relative number of citations (compared to tools in the same benchmarks) and the maximum H and M indices for the corresponding authors. The results are presented in Figure 1A. We found significant associations between most of the citation-based metrics (journal H5, IF, citations, relative citations, H-index and M-index). There is also a strong association between the year of publication, the relative age and many of the citation-based metrics.

We found that author reputation metrics, journal impacts and the age of methods were **not** significant predictors of either method accuracy or speed (see Figure 1B for the associations with accuracy). The strongest association was between accuracy and journal impact factor (Spearman's ρ = -0.1, P-value = 0.16). Linear mixed models of these parameters and accuracy also failed to identify an association between these (accuracy: R^2 = 0.03, P-value = 0.81; speed: R^2 = 0.04, P-value = 0.66). We computed a correlations between speed and accuracy for each benchmark and used weighted sum Z-tests (90) to further investigate for an association between speed and accuracy. This also failed to identify a significant relationship (sum Z=-0.1, P-value=0.5).

To investigate further the association between speed and accuracy, we ran a 10,000-fold Monte Carlo permutation test. The results of which are shown in a 10x10 grid in Figure 1C. We identified 21 bins where there was a significant excess or dearth of methods. There was an excess of “slow and inaccurate” software (Z=1.6, P-value=0.05) and “slow and accurate” software (Z=1.9, P-value=0.03). We find that the amount of software classed as “fast and accurate” and “fast and inaccurate” are at roughly the expected numbers. The number of significant results is not simply an excess of false-positives from multiple testing, as the probability of finding 21 of 100 tests significant by chance is very low (91).

The most significant finding from this analysis is that the number of software tools that are classed as intermediate in terms of both speed and accuracy is very much underrepresented in the four central cells highlighted in Figure 1C (Z = -1.7, -2.1, -2.1 and -2.4, P-values = 0.04, 0.02, 0.02 and 0.008, respectively, reading from top to bottom, left to right). We also tested relative age of the “slow and inaccurate” methods, these were generally published earlier than other methods (W = 31, P=0.007, one-tailed Wilcoxon test).

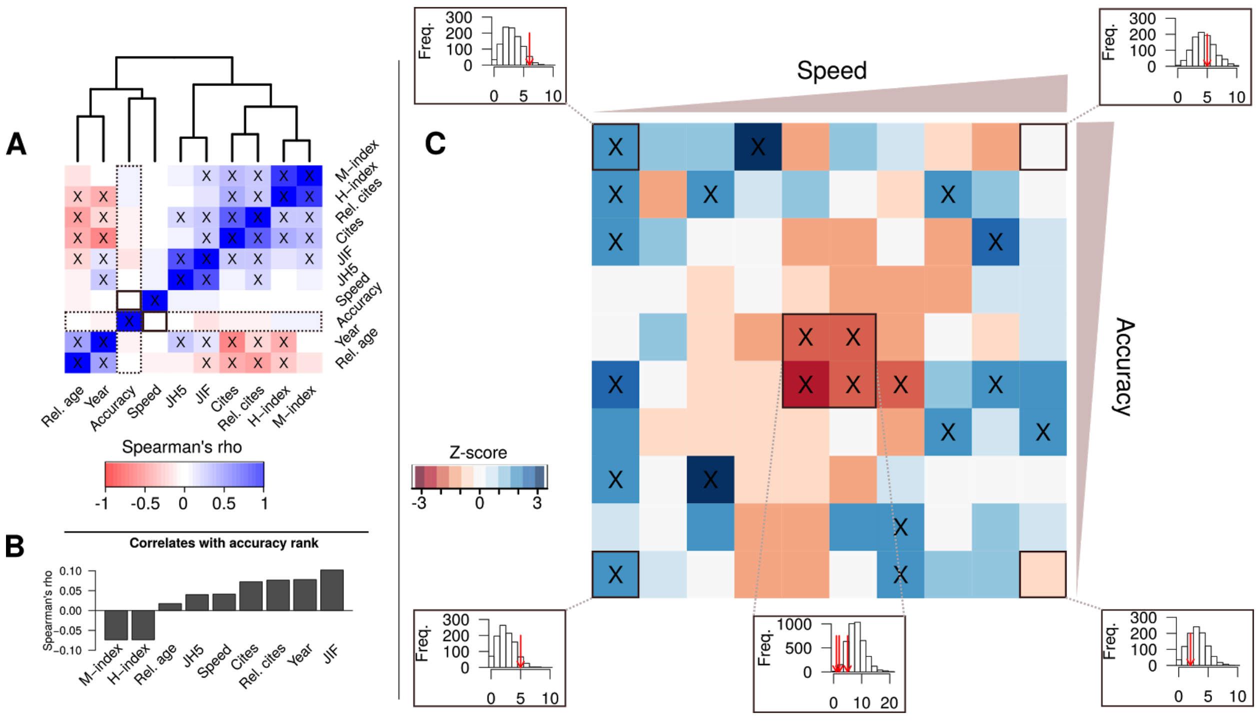

Figure 1: A. A heatmap indicating the relationships between proposed predictors of software quality. Spearman’s rho is used to infer correlations between metrics such as the H and M indices of corresponding authors, number of citations, journal impact factors and H5 indices,

the year and relative age of software and the mean relative rankings of software speed and accuracy. Correlations with a P-value less than 0.05 are indicated with a ‘X’. The dashed rectangular area is illustrated in more detail in **B**, the bold square is shown in more detail in **C**. **B.** A barplot illustrating the correlation as measured by Spearman’s rho between potential predictors and mean normalised accuracy ranks in more detail. **C.** A heatmap indicating the relative paucity or abundance of software in a the range of possible accuracy and speed rankings. Blue colours indicate an abundance of software tools in an accuracy and speed category, while red 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 10,000 random permutations of the speed and accuracy ranks from each benchmark. The accuracy and speed mean normalised ranks have been binned into 100 classes (a 10x10 grid) that range from comparatively slow/inaccurate to comparatively fast/accurate. Z-scores with a P-value less than 0.05 are indicated with a ‘X’.

Discussion

We have gathered data on the relative speed and accuracies of 248 bioinformatic methods from 43 benchmarks that were published between 2005 and 2016. The most dramatic result from this work is that there is a **major under-representation of software that has both intermediate levels of accuracy and speed**. This strongly suggests that bioinformatic software tools suffer from a form of **publication bias**. Our community of developers, reviewers and possibly editors appear to be unwilling to publish tools of this genre. This is unfortunate, since problems that lack fast and accurate solutions force researchers to make unnecessary compromises in terms of accuracy and time. If our hypothesis that this underrepresentation is due to publication bias then why is there an enrichment of slow and inaccurate software (P-value=0.05, empirical distributions from permutation tests)? How could these methods be published? It appears that these tools are generally published earlier than competing method (P=0.007, one-tailed Wilcoxon test), therefore comparisons were not required to publish these methods.

We found that there are no significant predictors of software accuracy. Neither, author reputation, number of citations, journal impact, relative age or speed appear to be significant predictors of whether a software tool is accurate. Linear mixed models of these values also fail to identify predictors of software accuracy, as do methods for combining P-values (92) [REFERENCE & NOMENCLATURE!!!]. The poor relationship between accuracy and both author reputation and number of citations is particularly troubling, since both are related to “word of mouth” and “previously used in a similar analysis” which a recent survey of researchers suggested is a major influence on the tools are selected (37). This implies that the recorded high citation rates for bioinformatic software (1–3) is more a reflection of user-friendliness and the Matthew Effect (93, 94).

The lack of any relationship between software speed and accuracy is particularly surprising. The slow software tools are found to be overrepresented at both high and low levels of accuracy (Figure 1C). Likewise accurate tools were found at both high and low speed ranges. A simple gedankenexperiment is sufficient to prove that slow software is likely to be less thoroughly tested. Since typical software development is an iterative process, where methods are refined over successive rounds of testing and evaluation. As a consequence slow tools undergo fewer testing cycles than fast methods (assuming similar time-spans are spent on most projects e.g. the span of a M.Sc. or Ph.D.). This implies that much of computational biology software is developed using sluggish waterfall or spiral methods, as opposed to using agile methods [REFERENCES?]. Supporting this idea is the fact that fast and very inaccurate software is relatively rare, implying that faster methods undergo additional testing and refinement.

We note that the speed of an algorithm may be influenced by factors that are independent of the algorithm choice, for example, compiled programming languages such as C usually outperform scripting languages such as Perl (43), operating system (43) and machine architecture (95, 96) and software implementation style may also be an influence (97). However, an $O(n^8)$ algorithm will scale poorly, irrespective of architecture, language or programming style. While a $O(n^2)$ method scale well. Therefore we think that language and architecture choice won't have a major influence.

Conclusions

The biological sciences is increasingly a data-driven science, therefore the dependence of biologists on software is also increasing (22). The commensurate increases in the complexity of software tools creates an increased likelihood that software bugs are introduced (98). Scientific progress is made by testing hypotheses using appropriate positive and negative controls, independent experimental tests and replication (99, 100). We think that software and analysis tools should be no exception to these principles. Scientific software should be thoroughly tested by developers, however, the results of developer and author-derived tests should treated with caution (26).

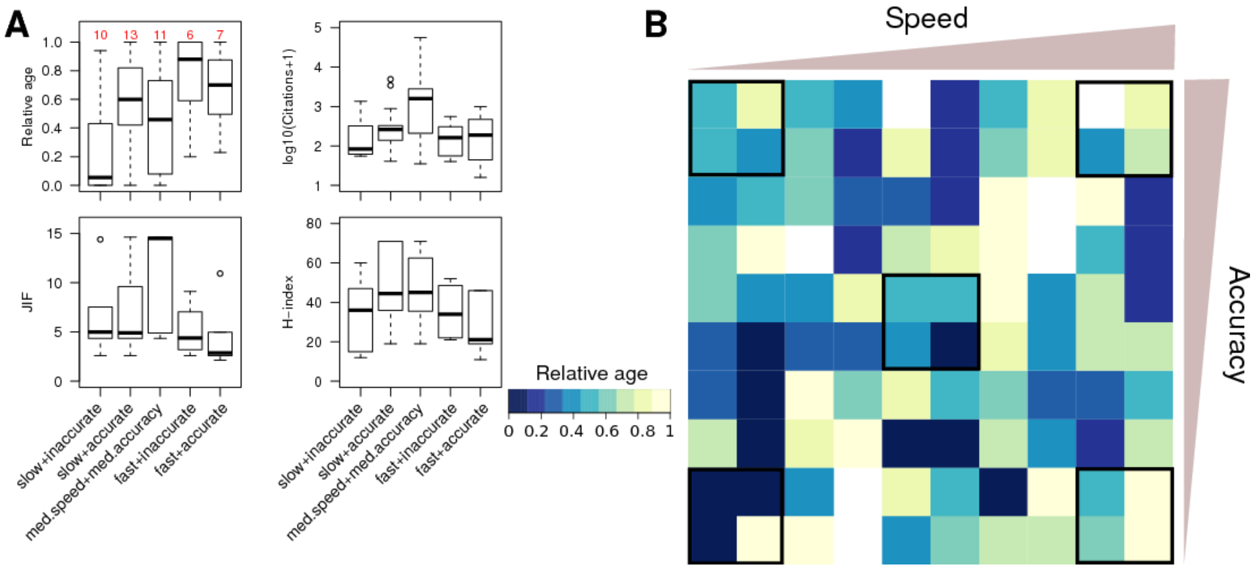

Figure 2: A. Box and whisker plots for relative age, number of citations, journal impact factor (JIF) and the H-index of corresponding authors. The four boxes correspond to the four extreme corners of the speed vs accuracy spectrum (i.e. slow and inaccurate, slow and accurate, fast and inaccurate, fast and accurate). **B.** A heatmap indicating the relative age of software in a range of relative accuracy and speed rankings. Blue colours indicate an abundance of older software tools in an accuracy and speed category, while light colours indicate younger software in an accuracy and speed category.

Our analysis has given some interesting glimpses into the dynamics of software development and publication systems. We have found that slow and inaccurate software is typically published early in the development of a field (see Figure 2). However as more software packages become available inaccurate tools become more difficult to publish. However, software tools do not uniformly span the space of speed and accuracy. We observe that trade-offs between speed and accuracy are made, typically accuracy seems to be favoured over speed, however there is a significant gap left in software-space, there is a significant under-abundance of software that is intermediate in terms of speed and accuracy. This points to an unfortunate publication bias in computational biology software. These classes of tools may prove to be useful pre-filters for more accurate approaches or could be used when datasets are too large to processed when the only accurate tools are slow (e.g. (101, 102)).

We propose that the full spectrum of algorithm accuracies and speeds may serve a useful purpose. Ideally all tools are robust, well tested, accurate and fast. However, those tools that are not, may yet serve a useful purpose. Like negative results, if honestly reported, illustrate to the research community that certain approaches are not useful research avenues (99, 103, 104). Ideally the main publishers of computational biology tools should be aware that their behaviour may unduly influence the dynamics of software development. Of the 248 methods we have used in this study, the top 5 journals are Bioinformatics (65 tools), Nucleic Acids Research (21 tools), Genome Research (21 tools), BMC Bioinformatics (20 tools) and the Journal of Molecular Biology (10 tools).

A potential avenue for further exploration is to compare the starting point for software development projects as certain approaches may produce more rapid gains than others. For example, starting with biologically plausible and fast methods will theoretically allow more rapid gains in accuracy through iterative method refinement than starting with slow and mathematically complete approaches. At the very least these may be used as rapid data filters for reducing the size and complexity of problems, prior to employing more rigorous methods.

We have shown that accurate software is not necessarily the most recently released method or the product of high profile lab groups or selected by high impact journals. Software that is widely used or is either slow or fast is also not necessarily the most accurate. Therefore, accurate software may be the product of features we have not been able to capture. Possibly, hard work, good ideas and sound method testing.

Finally, we think that the field of computational biology could benefit from embracing an increased number of independent software tests (88) as well as increase the use of independent implementations of software to improve robust conclusions (98)**.** This will reduce the over-optimistic and misleading reporting of method accuracy (24–26).

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 Anne-Laure Boulesteix's (ALB) three criteria for a "neutral comparison study" (88). Firstly, the main focus of the article is the comparison and **not** the introduction of a new method, secondly, the authors should be reasonably neutral and thirdly, the test data and evaluation criteria should be sensible.

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 (47, 105)). 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

In order to identify a list of high frequency (e.g. *f(word)* > 1/10,000) words by pooling the content of two control texts (106, 107).
Secondly, in order to compute a log-odds score for bioinformatic words, we computed the frequency of words that passed 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 ((47, 105) to select 8,908 articles that match the word "bioinformatics" in the title or abstract and were published between 2013 and 2015. We computed word frequencies for each non-high frequency words by combining text from titles and abstracts for the background and training articles. A log-odds score is computed for each word using the following formula: $lo(w) = log_2(\frac{f_{tr}(word) + \delta}{f_{bg}(word) + \delta})$, where δ is a prior probability ($\delta = 10^{-5}$, by default), $f_{bg}(word)$ and $f_{tr}(word)$ are the frequencies of a $word$ in the background and training datasets respectively. Word frequencies are computed by counting the number of times a word appears in the pool of titles and abstracts, the counts are 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 are likely to be benchmarks of bioinformatic tools, these may 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 are extracted from the articles that meet the criteria, and these are also added to the set of training articles. The evaluated candidate articles that do not meet the ALB-criteria are incorporated into the set of background articles.
This process is iterated a number of times and has resulted in the identification of **43** benchmark articles, that contain **102** different benchmarks, together these rank **248** distinct software packages.

Data extraction: 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 method. 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 method. In line with rank-based statistics, the cases where methods were tied are resolved by using a midpoint rank (e.g. if method 3 and 4 are tied, the rank 3.5 is used) (108). Each rank extraction was independently verified by at least one other co-author to ensure both the provence 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 $\frac{r-1}{n-1}$ where '*r*' is a method's rank and '*n*' is the number of methods in the benchmark. For methods that were benchmarked multiple times (e.g. BWA is evaluated in 6 different articles (53, 54, 56–58, 60)) a mean normalised rank is used to summarise the performance for methods tested multiple times with multiple metrics.
For each method we identified the corresponding publications in GoogleScholar, the total number of citations was recorded, the corresponding authors were also identified and if these 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. For the journals that each method is published in we extracted the "journal impact factor" (JIF) and the H5-index from Thompson-Reuters and GoogleScholar Metrics databases. The year of publication was also recorded for each method. A "relative age" and "relative citations" was also computed for each method. For each benchmark, tools were ranked by year of first publication (or number of

citations), ranks were assigned and then normalised as described above. Methods ranked in multiple evaluations were then assigned a mean value for "relative age" and "relative citations".

Statistical analysis: The 10 statistics that we have gathered for each method were evaluated in a pairwise fashion to produce Figure 1A&B, the R code for these is given in the supplement. For each set of method ranks we also generated 10,000 randomly generated permutations, each of which was used to produce summary statistics for each method as described above for the un-randomised data. These permuted rank statistics were used to generate Figure 1C. For each cell in a 10x10 grid a Z-score ($Z = \frac{x-\bar{x}}{\sigma}$) is computed to illustrate the abundance or lack of methods in a cell relative to the permuted data.

Data availability:

The raw datasets are available here: https://docs.google.com/spreadsheets/d/14xIY2PHNvxmV9MQLpbzSfFkuy1RlzDHbBOCZLJKcGu8/edit?usp=sharing

Additional documentation, code, figures and raw data is available here: https://github.com/UCanCompBio/speed-vs-accuracy-meta-analysis

Acknowledgments

The authors acknowledge the contribution of invaluable discussions with Shinichi Nakagawa, Suetonia Palmer and Jason Tylianakis.

References

1. Perez-Iratxeta C, Andrade-Navarro MA, Wren JD (2007) Evolving research trends in bioinformatics. *Brief Bioinform* 8(2):88–95.

2. Van Noorden R, Maher B, Nuzzo R (2014) The top 100 papers. *Nature* 514(7524):550–553.

3. Wren JD (2016) Bioinformatics programs are 31-fold over-represented among the highest impact scientific papers of the past two decades. *Bioinformatics*. doi:10.1093/bioinformatics/btw284.

4. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. *J Mol Biol* 215(3):403–410.

5. Thompson JD, Higgins DG, Gibson TJ (1994) 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.

6. Thompson JD, Gibson TJ, Plewniak F, Jeanmougin F, Higgins DG (1997) The CLUSTAL_X

windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res* 25(24):4876–4882.

7. Altschul SF, et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 25(17):3389–3402.

8. Felsenstein J (1985) Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution* 39(4):783–791.

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

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

11. Ronquist F, Huelsenbeck JP (2003) MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics* 19(12):1572–1574.

12. Tamura K, Dudley J, Nei M, Kumar S (2007) MEGA4: Molecular Evolutionary Genetics Analysis (MEGA) software version 4.0. *Mol Biol Evol* 24(8):1596–1599.

13. Kaplan EL, Meier P (1958) Nonparametric Estimation from Incomplete Observations. *J Am Stat Assoc* 53(282):457–481.

14. Cox DR (1972) Regression models and life-tables. *J R Stat Soc Series B Stat Methodol*:187–220.

15. Sheldrick GM (1990) Phase annealing in SHELX-90: direct methods for larger structures. *Acta Crystallogr A* 46(6):467–473.

16. Sheldrick GM (2008) A short history of SHELX. *Acta Crystallogr A* 64(Pt 1):112–122.

17. Jones TA, Zou JY, Cowan SW, Kjeldgaard M (1991) 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.

18. Laskowski RA, MacArthur MW, Moss DS, Thornton JM (1993) PROCHECK: a program to check the stereochemical quality of protein structures. *J Appl Crystallogr* 26(2):283–291.

19. Otwinowski Z, Minor W (1997) [20] Processing of X-ray diffraction data collected in oscillation mode. *Methods in Enzymology* (Academic Press), pp 307–326.

20. Kraulis PJ (1991) MOLSCRIPT: a program to produce both detailed and schematic plots of protein structures. *J Appl Crystallogr* 24(5):946–950.

21. Berman HM, et al. (2000) The Protein Data Bank. *Nucleic Acids Res* 28(1):235–242.

22. Marx V (2013) Biology: The big challenges of big data. *Nature* 498(7453):255–260.

23. Gombiner J (2011) Carbon footprinting the internet. *Consilience-The Journal of Sustainable Development* 5(1). Available at:

http://www.consiliencejournal.org/index.php/consilience/article/viewFile/141/57.

24. Boulesteix A-L (2010) Over-optimism in bioinformatics research. *Bioinformatics* 26(3):437–439.

25. Jelizarow M, Guillemot V, Tenenhaus A, Strimmer K, Boulesteix A-L (2010) Over-optimism in bioinformatics: an illustration. *Bioinformatics* 26(16):1990–1998.

26. Norel R, Rice JJ, Stolovitzky G (2011) The self-assessment trap: can we all be better than average? *Mol Syst Biol* 7(1):537.

27. Egan JP (1975) *Signal Detection Theory and ROC-analysis* (Academic Press, New York).

28. Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Trans Software Eng* 38(6):1276–1304.

29. Bujnicki JM, Elofsson A, Fischer D, Rychlewski L (2001) LiveBench-1: continuous benchmarking of protein structure prediction servers. *Protein Sci* 10(2):352–361.

30. Puton T, Kozlowski LP, Rother KM, Bujnicki JM (2014) CompaRNA: a server for continuous benchmarking of automated methods for RNA secondary structure prediction. *Nucleic Acids Res* 42(8):5403–5406.

31. Barton M nucleotides · genome assembler benchmarking. Available at: http://nucleotid.es/ [Accessed December 18, 2015].

32. Felsenstein J (1995) Phylogeny programs. *Internet address: http://evolution gs washington edu/phylip/software html*. Available at: http://evolution.gs.washington.edu/phylip/software.html.

33. Altschul S, et al. (2013) The anatomy of successful computational biology software. *Nat Biotechnol* 31(10):894–897.

34. Henry VJ, Bandrowski AE, Pepin A-S, Gonzalez BJ, Desfeux A (2014) OMICtools: an informative directory for multi-omic data analysis. *Database* 2014. doi:10.1093/database/bau069.

35. Wikipedia contributors (2015) List of sequence alignment software. *Wikipedia, The Free Encyclopedia*. Available at: https://en.wikipedia.org/w/index.php?title=List_of_sequence_alignment_software&oldid=693586242 [Accessed December 18, 2015].

36. Wikipedia contributors (2015) List of RNA structure prediction software. *Wikipedia, The Free Encyclopedia*. Available at: https://en.wikipedia.org/w/index.php?title=List_of_RNA_structure_prediction_software&oldid=693718881 [Accessed December 18, 2015].

37. Loman N, Connor T (2015) Bioinformatics infrastructure and training survey. doi:10.6084/M9.FIGSHARE.1572287.V2.

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

39. Woolley AW, Chabris CF, Pentland A, Hashmi N, Malone TW (2010) Evidence for a collective intelligence factor in the performance of human groups. *Science* 330(6004):686–688.

40. Cheruvelil KS, et al. (2014) Creating and maintaining high-performing collaborative research teams: the importance of diversity and interpersonal skills. *Front Ecol Environ* 12(1):31–38.

41. Hirsch JE (2005) An index to quantify an individual's scientific research output. *Proc Natl Acad Sci U S A* 102(46):16569–16572.

42. Bornmann L, Mutz R, Daniel H-D (2008) 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.

43. Fourment M, Gillings MR (2008) A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics* 9:82.

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

45. Papadimitriou CH Computational Complexity. *Encyclopedia of Computer Science* (John Wiley and Sons Ltd., Chichester, UK), pp 260–265.

46. Wadi L, Meyer M, Weiser J, Stein LD, Reimand J (2016) *Impact of knowledge accumulation on pathway enrichment analysis* doi:10.1101/049288.

47. Sayers EW, et al. (2010) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* 38(Database issue):D5–16.

48. Freyhult EK, Bollback JP, Gardner PP (2007) Exploring genomic dark matter: a critical assessment of the performance of homology search methods on noncoding RNA. *Genome Res* 17(1):117–125.

49. Jünemann S, et al. (2014) GABenchToB: a genome assembly benchmark tuned on bacteria and benchtop sequencers. *PLoS One* 9(9):e107014.

50. Tran H, Porter J, Sun M-A, Xie H, Zhang L (2014) Objective and comprehensive evaluation of bisulfite short read mapping tools. *Adv Bioinformatics* 2014:472045.

51. Zhang W, et al. (2011) A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PLoS One* 6(3):e17915.

52. Abbas MM, Malluhi QM, Balakrishnan P (2014) Assessment of de novo assemblers for draft genomes: a case study with fungal genomes. *BMC Genomics* 15 Suppl 9:S10.

53. Bao S, et al. (2011) Evaluation of next-generation sequencing software in mapping and

assembly. *J Hum Genet* 56(6):406–414.

54. Caboche S, Audebert C, Lemoine Y, Hot D (2014) Comparison of mapping algorithms used in high-throughput sequencing: application to Ion Torrent data. *BMC Genomics* 15:264.

55. Kleftogiannis D, Kalnis P, Bajic VB (2013) Comparing memory-efficient genome assemblers on stand-alone and cloud infrastructures. *PLoS One* 8(9):e75505.

56. Hatem A, Bozdağ D, Toland AE, Çatalyürek ÜV (2013) Benchmarking short sequence mapping tools. *BMC Bioinformatics* 14:184.

57. Schbath S, et al. (2012) Mapping reads on a genomic sequence: an algorithmic overview and a practical comparative analysis. *J Comput Biol* 19(6):796–813.

58. Ruffalo M, LaFramboise T, Koyutürk M (2011) Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics* 27(20):2790–2796.

59. Yang X, Chockalingam SP, Aluru S (2013) A survey of error-correction methods for next-generation sequencing. *Brief Bioinform* 14(1):56–66.

60. Holtgrewe M, Emde A-K, Weese D, Reinert K (2011) A novel and well-defined benchmarking method for second generation read mapping. *BMC Bioinformatics* 12:210.

61. Rackham OJL, Dellaportas P, Petretto E, Bottolo L (2015) WGBSSuite: simulating whole-genome bisulphite sequencing data and benchmarking differential DNA methylation analysis tools. *Bioinformatics* 31(14):2371–2373.

62. Huang HW, NISC Comparative Sequencing Program, Mullikin JC, Hansen NF (2015) Evaluation of variant detection software for pooled next-generation sequence data. *BMC Bioinformatics* 16:235.

63. Thompson JD, Linard B, Lecompte O, Poch O (2011) A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS One* 6(3):e18093.

64. Nuin PAS, Wang Z, Tillier ERM (2006) The accuracy of several multiple sequence alignment programs for proteins. *BMC Bioinformatics* 7:471.

65. Pais FS-M, Ruy P de C, Oliveira G, Coimbra RS (2014) Assessing the efficiency of multiple sequence alignment programs. *Algorithms Mol Biol* 9(1):4.

66. Pervez MT, et al. (2014) Evaluating the accuracy and efficiency of multiple sequence alignment methods. *Evol Bioinform Online* 10:205–217.

67. Liu K, Linder CR, Warnow T (2010) Multiple sequence alignment: a major challenge to large-scale phylogenetics. *PLoS Curr* 2:RRN1198.

68. Maška M, et al. (2014) A benchmark for comparison of cell tracking algorithms. *Bioinformatics* 30(11):1609–1617.

69. Li Y, et al. (2012) Performance comparison and evaluation of software tools for microRNA

deep-sequencing data analysis. *Nucleic Acids Res* 40(10):4298–4305.

70. Lu B, Zeng Z, Shi T (2013) Comparative study of de novo assembly and genome-guided assembly strategies for transcriptome reconstruction based on RNA-Seq. *Sci China Life Sci* 56(2):143–155.

71. Liu R, Loraine AE, Dickerson JA (2014) Comparisons of computational methods for differential alternative splicing detection using RNA-seq in plant systems. *BMC Bioinformatics* 15:364.

72. Kumar S, Vo AD, Qin F, Li H (2016) Comparative assessment of methods for the fusion transcripts detection from RNA-Seq data. *Sci Rep* 6:21597.

73. Pain A, et al. (2015) An assessment of bacterial small RNA target prediction programs. *RNA Biol* 12(5):509–513.

74. Tikk D, Thomas P, Palaga P, Hakenberg J, Leser U (2010) A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature. *PLoS Comput Biol* 6:e1000837.

75. Kolodny R, Koehl P, Levitt M (2005) Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures. *J Mol Biol* 346(4):1173–1188.

76. Wallner B, Elofsson A (2005) All are not equal: a benchmark of different homology modeling programs. *Protein Sci* 14(5):1315–1327.

77. Shang J, et al. (2011) Performance analysis of novel methods for detecting epistasis. *BMC Bioinformatics* 12:475.

78. Lindgreen S, Adair KL, Gardner PP (2016) An evaluation of the accuracy and speed of metagenome analysis tools. *Sci Rep* 6:19233.

79. Bazinet AL, Cummings MP (2012) A comparative evaluation of sequence classification programs. *BMC Bioinformatics* 13:92.

80. Saha S, Bridges S, Magbanua ZV, Peterson DG (2008) Empirical comparison of ab initio repeat finding programs. *Nucleic Acids Res* 36(7):2284–2294.

81. Lange E, Tautenhahn R, Neumann S, Gröpl C (2008) Critical assessment of alignment procedures for LC-MS proteomics and metabolomics measurements. *BMC Bioinformatics* 9:375.

82. Yang C, He Z, Yu W (2009) Comparison of public peak detection algorithms for MALDI mass spectrometry data analysis. *BMC Bioinformatics* 10:4.

83. Liu K, Linder CR, Warnow T (2011) RAxML and FastTree: comparing two methods for large-scale maximum likelihood phylogeny estimation. *PLoS One* 6(11):e27731.

84. Yang J, Warnow T (2011) Fast and accurate methods for phylogenomic analyses. *BMC Bioinformatics* 12 Suppl 9:S4.

85. Oscamou M, et al. (2008) Comparison of methods for estimating the nucleotide substitution matrix. *BMC Bioinformatics* 9:511.

86. Bayzid MS, Warnow T (2013) Naive binning improves phylogenomic analyses. *Bioinformatics* 29(18):2277–2284.

87. Liu K, Nelesen S, Raghavan S, Linder CR, Warnow T (2009) Barking up the wrong treelength: the impact of gap penalty on alignment and tree accuracy. *IEEE/ACM Trans Comput Biol Bioinform* 6(1):7–21.

88. Boulesteix A-L, Lauer S, Eugster MJA (2013) A plea for neutral comparison studies in computational sciences. *PLoS One* 8(4):e61562.

89. Garfield E (2006) The history and meaning of the journal impact factor. *JAMA* 295(1):90–93.

90. Zaykin DV (2011) Optimally weighted Z-test is a powerful method for combining probabilities in meta-analysis. *J Evol Biol* 24(8):1836–1841.

91. Moran MD (2003) Arguments for Rejecting the Sequential Bonferroni in Ecological Studies. *Oikos* 100(2):403–405.

92. Chambers JM, Hastie T (1992) *Statistical Models in S* (Wadsworth & Brooks/Cole Advanced Books & Software).

93. Merton RK, Others (1968) The Matthew effect in science. *Science* 159(3810):56–63.

94. Larivière V, Gingras Y (2010) The impact factor's Matthew Effect: A natural experiment in bibliometrics. *J Am Soc Inf Sci* 61(2):424–427.

95. Rognes T, Seeberg E (2000) Six-fold speed-up of Smith–Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics* 16(8):699–706.

96. Farrar M (2007) Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics* 23(2):156–161.

97. Briand LC, Wüst J, Daly JW, Victor Porter D (2000) Exploring the relationships between design measures and software quality in object-oriented systems. *J Syst Softw* 51(3):245–273.

98. Darriba D, Flouri T, Stamatakis A (2015) The State of Software in Evolutionary Biology. *bioRxiv*:031930.

99. Ioannidis JPA (2005) Why most published research findings are false. *PLoS Med* 2(8):e124.

100. Moonesinghe R, Khoury MJ, A Cecile J (2007) Most Published Research Findings Are False—But a Little Replication Goes a Long Way. *PLoS Med* 4(2):e28.

101. Lowe TM, Eddy SR (1997) tRNAscan-SE: a program for improved detection of transfer

RNA genes in genomic sequence. *Nucleic Acids Res* 25(5):955–964.

102. Weinberg Z, Ruzzo WL (2006) Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics* 22(1):35–39.

103. Workman C, Krogh A (1999) No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. *Nucleic Acids Res* 27(24):4816–4822.

104. Rivas E, Eddy SR (2000) Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics* 16(7):583–605.

105. McEntyre J, Lipman D (2001) PubMed: bridging the information gap. *CMAJ* 164(9):1317–1319.

106. Carroll L (1865) *Alice's adventures in Wonderland* (Macmillan and Co., London).

107. Tolkien JRR (1937) *The Hobbit, Or, There and Back Again* (George Allen & Unwin, UK).

108. Mann HB, Whitney DR (1947) On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Ann Math Stat* 18(1):50–60.