Large-scale comparative bioinformatics analyses

Maria Chatzou

TESI DOCTORAL UPF / ANY 2016

DIRECTOR DE LA TESI

Cedric Notredame
Bioinformatics and Genomics
Center for Genomic Regulation

Department of Experimental and Health Sciences

Acknowledgements

Journeys are made by the people you travel with and my PhD journey has been an incredible one because of the people in it. So here I would like to take the opportunity to thank these incredible people.

It's rare in life that you get to work with a great scientist and it is even more rare that you get to work with both a great scientist and a great mentor. I have had the privilege to be supervised by both a great scientist and a great mentor, Cedric Notredame. So thank you very much Cedric for all the support, help and wisdom you provided me during the 4 years of my PhD journey. I would have never evolved to the scientist and the person I am without your constant mentoring.

Working in a group of very talented and smart people I soon realized that the most important bit it's not just having great ideas but having great people dynamics. I have been very lucky to be part of not just a group of people, but of an excellent, very positive and creative team. For this and for being great colleagues and above all great friends I would really like to thank all my team members Pablo Prieto, Paolo Di Tommaso, Cedrik Magis, Jose Espinosa, Ionas Erb, Evan Floden, Carsten Kemena and Jia-Ming Chang.

Moreover I would like to thank Martin Steinegger for all of his help and his friendship, as well as Johannes Soding for great brainstorming discussions, and both of them, along with Maria Hauser, for making my stay at Max Planck Institute for Biophysical Chemistry in Munich, a great and very productive one.

I also wish to thank Olivier Lichtarge for warmly welcoming me to work with him and his group at Baylor College of Medicine, in Houston, Texas. Special thanks go to Angela Wilkins and Panos Katsonis for the great time I had discussing and working with them. Along with Olivier they made my time in Houston unforgettable and our collaboration one looking forward to resume in the future.

Further thanks go to my thesis committee members, Ben Lehner, Fyodor Kondrashov and Mar Alba, who also accompanied me on my PhD way. I am also very grateful to Olivier Gascuel and Roderic Guigo for all the attention they paid to my work and for the ideas and insightful comments they offered. A big "thank you" should also go to Romina Garrido for always taking great care of all the organization aspects.

Here I would especially like to thank my sister Theopisti Chatzou for all her support and love. She has been there to help me when needed the most and provide me with the strength and confidence needed to continue without losing my excitement.

Finally I wish to thank all my friends and family for their continuous support and belief in my abilities to become one day a great scientist and an even greater person.

“ ***If you want to go fast, go alone. If you want to go far, go together*****. ”**

— African Proverb

Abstract

One of the main and most recent challenges of modern biology is to keep-up with growing amount of biological data coming from next generation sequencing technologies. Keeping up with the growing volumes of experiments will be the only way to make sense of the data and extract actionable biological insights. Large-scale comparative bioinformatics analyses are an integral part of this procedure. When doing comparative bioinformatics, multiple sequence alignments (MSAs) are by far the most widely used models as they provide a unique insight into the accurate measure of sequence similarities and are therefore instrumental to revealing genetic and/or functional relationships among evolutionarily related species. Unfortunately, the well-established limitation of MSA methods when dealing with very large datasets potentially compromises all downstream analyses. In this thesis I expose the current relevance of multiple sequence aligners, I show how their current scaling up is leading to serious numerical stability issues and how they impact phylogenetic tree reconstruction. For this purpose, I have developed two new methods, MEGA-Coffee, a large scale aligner and Shootstrap a novel bootstrapping measure incorporating MSA instability with branch support estimates when computing trees. The large amount of computation required by these two projects was carried using Nextflow, a new computational framework that I have developed to improve computational efficiency and reproducibility of large-scale analyses like the ones carried out in the context of these studies.

Resumen

Uno de los principales y más recientes retos de la biología moderna es poder hacer frente a la creciente cantidad de datos biológicos procedentes de las tecnologías de secuenciación de alto rendimiento. Mantenerse al día con los crecientes volúmenes de datos experimentales es el único modo de poder interpretar estos datos y extraer conclusiones biológicos relevantes. Los análisis bioinformáticos comparativos a gran escala son una parte integral de este procedimiento. Al hacer bioinformática comparativa, los alineamientos múltiple de secuencias (MSA) son con mucho los modelos más utilizados, ya que proporcionan una visión única de la medida exacta de similitudes de secuencia y son, por tanto, fundamentales para inferir las relaciones genéticas y / o funcionales entre las especies evolutivamente relacionadas. Desafortunadamente, la conocida limitación de los métodos MSA para analizar grandes bases de datos, puede potencialmente comprometer todos los análisis realizados a continuación. En esta tesis expongo la relevancia actual de los métodos de alineamientos multiples de secuencia, muestro cómo su uso en datos masivos está dando lugar a serios problemas de estabilidad numérica y su impacto en la reconstrucción del árbol filogenético. Para este propósito, he desarrollado dos nuevos métodos, MEGA-café, un alineador de gran escala y Shootstrap una nueva medida de bootstrapping que incorpora la inestabilidad del MSA con las estimaciones de apoyo de rama en el cálculo de árboles filogéneticos. La gran cantidad de cálculo requerido por estos dos proyectos se realizó utilizando Nextflow, un nuevo marco computacional que se ha desarrollado para mejorar la eficiencia computacional y la reproducibilidad del análisis a gran escala como la que se lleva a cabo en el contexto de estos estudios.

Preface

The fast evolution of sequencing technologies combined with the exponential decrease in the cost of sequencing has resulted in an unprecedented growth of biological data. Such a volume of data is challenging traditional analysis procedures, which were typically based on the add-hoc combination of simple scripts running on a personal computer (PC). Bioinformaticians are now increasingly required to use high performance computing (HPC) and big data technologies to do even the simplest analysis. These new approaches bear increasing resemblance with the techniques developed by Google, Amazon, Facebook and other big-data companies. There is no sign that this trend may fade anytime soon, and the personalized medicine objectives, such as those laid out in the Obama Precision Medicine Initiative (PMI) suggest that large-scale analyses currently carried out in research environment will soon need to be deployed in production mode by hospital and health workers. This scaling up raises many important issues, the most pressing ones being reliability and reproducibility. The purpose of this thesis has been to explore the impact of large-scale data analysis on multiple sequence alignment and phylogenetic reconstruction, two of the most popular modeling methods in biology. The instability we have found may have important consequences for personalized medicine, owing to the growing importance of these methods in treatments relying on personalized genomics analyses. In this thesis I develop these issues and their consequences, I introduce the solutions I have developed, in the form of a new generation of computational tools allowing efficient and reproducible computation of complex pipeline based analyses.

Contents

Objectives

The objectives of this PhD thesis are: 1) To critically assess Multiple Sequence Alignment (MSA) methods and the algorithmic frameworks they are based upon. 2) To quantify the effect that large-scale aligners instability has on downstream analyses. We have focused this quantification on structural and evolutionary analysis. 3) To design a new large-scale aligner able to efficiently deliver high quality alignments of datasets containing hundred and thousand of sequences. 4) To effectively address the two major computational issues related the large-scale analyses: computational irreproducibility and the need for high performance computing (HPC).

Chapter 1 is an introductory chapter that provides an exhaustive review of the development of multiple sequence alignment methods and their main applications. It is focused on the progress made over the past decade. It concludes that the main challenge for multiple sequence aligners will be to keep up with growing data set sizes.

This raises concerns on the ability of existing large-scale aligners to deliver high quality alignments of datasets containing thousands of sequences. These concerns led to the work described in Chapter 2, where we show that the methods used to compute large-scale models incorporating over 100 sequences are numerically unstable and very sensitive to alignment uncertainty. We demonstrate that this instability results in equally unstable phylogenetic trees. We, then, quantify this effect and propose a novel bootstrap method, shootstrap, which estimates the combined effect of alignment uncertainty and evolutionary sampling on phylogenetic tree branch supports.

Chapter 3 addresses the problem of improving multiple sequence aligner accuracy when dealing with large datasets. It introduces MEGA-Coffee a new alignment method able to simultaneously align any number of protein sequences in a quick and accurate manner, while minimizing alignment uncertainty and thus delivering much more stable alignments.

Chapter 4 is dedicated to computational reproducibility. To perform most of the analyses described in these chapters complex pipelines, with many dependencies on external scripts, binaries and libraries, had to be constructed. Furthermore, the computational demands were so high that HPC technologies had to be used (i.e. cluster, Amazon cloud, Barcelona Super Computer). At the same time computational reproducibility of the analyses was critical. We therefore developed a pipeline orchestration tool, Nextflow, specifically designed to address the reproducibility problem in computational pipelines while allowing researchers to easily write parallel and distributed data analysis applications in clusters and clouds.

1. Introduction - Multiple Sequence Alignment Modeling: Methods and Applications

Chatzou M, Magis C, Chang J-M, Kemena C, Bussotti G, Erb I, et al. Multiple sequence alignment modeling: methods and applications. Brief Bioinform. 2016 Nov;17(6):1009–23. DOI: 10.1093/bib/bbv099

1.1 Abstract

This review provides an overview on the development of Multiple Sequence Alignment (MSA) methods and their main applications. It is focused on progress made over the last decade. The three first sections review recent algorithmic developments for protein, RNA/DNA and genomic alignments. The fourth section deals with benchmarks and explores the relationship between empirical and simulated data, along with the impact on method developments. The last part of the review gives an overview on available MSA local reliability estimators and their dependence on various algorithmic properties of available methods.

2. Generalized bootstrap supports for protein sequences phylogenetic analyses incorporating alignment uncertainty

Chatzou M, Floden EW, Di Tommaso P, Gascuel O, Notredame C, Halanych K. Generalized Bootstrap Supports for Phylogenetic Analyses of Protein Sequences Incorporating Alignment Uncertainty. Halanych K, editor. Syst Biol. 2018 Mar 29; DOI: 10.1093/sysbio/syx096

2.1 Abstract

Phylogenetic reconstructions are essential in genomics data analyses and depend on accurate multiple sequence alignment (MSA) models. We show that all currently available large-scale progressive multiple alignment methods are numerically unstable and produce significantly different output when changing sequence input order. We used the HOMFAM protein sequences dataset to show that on datasets larger than 100 sequences, this instability affects on average 21% of the residues when considering the most stable aligners. The resulting Maximum Likelihood trees estimated from these multiple sequence alignments are equally unstable with over 38% of the branches being sensitive to the sequence input order. We established that about two-thirds of this uncertainty stems from unordered nature of children nodes within the guide trees used to estimate MSAs. To quantify this uncertainty we developed shootstrap, a novel approach that estimates the combined effect of alignment uncertainty and evolutionary sampling on phylogenetic tree branch supports. Compared to the regular bootstrap procedure, shootstrap provides a more informative support estimate.

3. MEGA-Coffee: Fast, accurate and stable large-scale alignments

Chatzou M, Kemena C, Steinegger M, Sodding J, Notredame C (2016). MEGA-Coffee: Fast, accurate and stable large-scale alignments. *In preparation*

3.1 Abstract

Multiple sequence alignment (MSA) is a critical step towards comparing sequence similarities and revealing genetic and/or functional relationships among evolutionarily related species. MEGA-Coffee is a novel and highly accurate method for producing stable and biological meaningful alignments of any number of sequences. MEGA-Coffee accuracy and scaling to large number of sequences are essential for coping with the ever-increasing information of biological databases and for better problem-driven data analyses, ultimately leading to higher-quality biological insights. MEGA-Coffee, outperforms existing alignment methods on large and/or difficult-to-align data sets in both terms of alignment quality and speed. At the same time MEGA-Coffee is less affected by alignment uncertainty compared to these methods. This opens up a new set of possibilities for users to utilize the information contained in large datasets, while re-enforcing reproducibility of data and biological interpretations.

3.2 Introduction

Multiple Sequence Alignment (MSA) is one of the most widely used bioinformatics methods in biology for the simultaneous comparison of functionally and/or evolutionarily related sequences [153] [1]. Usually they are an intermediate step towards more sophisticated applications such as phylogenetic reconstruction, profile estimation (often referred to as Hidden Markov Models, HMM), structural predictions, promoter analysis, active site identification, RNA secondary structure prediction, which are only some examples that depend on MSA. Building accurate MSAs is thus essential for these downstream analysis and biological applications to be correct.

For small data sets most recent alignment programs are able to compute accurate alignments in reasonable time. However the number of sequences to align is increasing rapidly and posing problems for current alignment algorithms. One well know issue is accuracy, and it has been extensively documented that the growing number of sequences has a negative impact on the accuracy of all the available heuristics [112] [14]. All these heuristics rely on the progressive algorithm and in this framework errors made early in the alignment process propagate and cause secondary alignment mistakes. More sequences means a higher number of initial errors and since each error has the potential to ruin the alignment, the probability of this happening becomes higher. This problem is further aggravated by the fact that methods like consistency [9], that have been shown to increase the accuracy of the resulting alignments scale poorly with the number of sequences. Other methods that make use of structural information suffer from similar problems. This is becoming more and more concerning as protein families with thousands of sequences are becoming more common as a result of various wide scale genome-sequencing projects.

Currently, the only methods that can routinely make alignments of more than about 10,000 sequences are Clustal-Omega [14], Mafft/PartTree [13] and the most recent ones Pasta [27] and Upp [154]. These methods are fast but efficiency comes at the cost of a sacrificed accuracy. Furthermore they are very sensitive to alignment uncertainty. In this paper, we introduce a new method, MEGA-Coffee, which allows alignments of very large sizes to be produced in an accurate and fast manner, while minimizing the effects of alignment uncertainty. The main ingredient of this accurate scaling up strategy is the guide tree. The most standard procedures for estimating a guide tree involve comparing all N sequences to each other, assigning time and memory requirements of $O(N^2)$. With MEGA-Coffee, we use a novel Agglomerative Hierarchical Greedy Clustering approach, StarClust, that has a complexity of $O(Nlog(N))$.

In a nutshell the MEGA-Coffee algorithm consists of three major steps: (I) filtering out the sequences that have up to 95% sequence identity, (II) clustering the remaining sequences utilizing

"StarClust", an agglomerative hierarchical greedy clustering algorithm and (III) aligning the clustered sequences. After the final alignment has been generated MEGA-Coffee offers the option to trim the aligned sequences in order to remove noise and increase alignment quality. We have used this new implementation to generate alignments of about 100,000 sequences in a manageable amount of time. In benchmark tests, our new algorithm can has shown to deliver MSAs of comparable accuracy to existing methods for small datasets (<100 seqs) and significantly more accurate than the most widely used methods for large datasets (>100 seqs).

3.3 Results

Table 1. HOMFAM benchmark results for different large-scale aligners. Accuracy achieved and its STDEV in Total Column scores (TC) are shown for different family size ranges. The average run time in minutes is also shown for every range.

	OVERALL 93 < N < 93,681 (94 families)			**SMALL** 93< N < 2,957 (41 families)			**MEDIUM** 3,127 < N < 9,105 (33 families)			**BIG** 10,099 < N < 93,681 (20 families)		
Aligner	TC (%)	STDEV (%)	Time (m)	TC (%)	STDEV (%)	Time (m)	TC (%)	STDEV (%)	Time (m)	TC (%)	STDEV (%)	Time (m)
MEGA-Coffee default	**66.98**	**0**	**4**	**72.16**	**0**	**0.5**	**70.06**	**0**	**4**	**51.29**	**0**	**12**
MEGA-Coffee noSync	66.92	2.74	4	71.73	2.28	0.5	70.72	2.88	4	50.80	3.48	12
Clustal-Omega	61.93	5.94	39	69.32	5.12	16	64.41	6.46	40	42.69	6.77	86
Upp	57.66	7.38	20	60.83	7.67	5	60.43	7.66	22	46.60	6.32	44
Pasta	56.78	7.03	41	60.70	7.41	7	58.19	6.74	38	46.44	6.74	113
Mafft	41.78	7.18	1	51.58	7.31	0.5	40.17	7.83	1	24.37	5.85	4

The standard method for measuring the accuracy of multiple alignment algorithms is to use a benchmark. A benchmark usually consists of test sequences with known structure and the corresponding reference alignments, generated using three-dimensional structural information. Owing to the lack of structures, large scale aligners are benchmarked using a small collection of structures with an available reference alignment. These sequences are then embedded within a larger set of sequences. Here we used Prefab [12] and BaliBase [110] benchmarks to quantify the quality of small MSAs (<100 sequences) generated by MEGA-Coffee and HOMFAM [15, 116] for larger MSAs (>100 sequences).

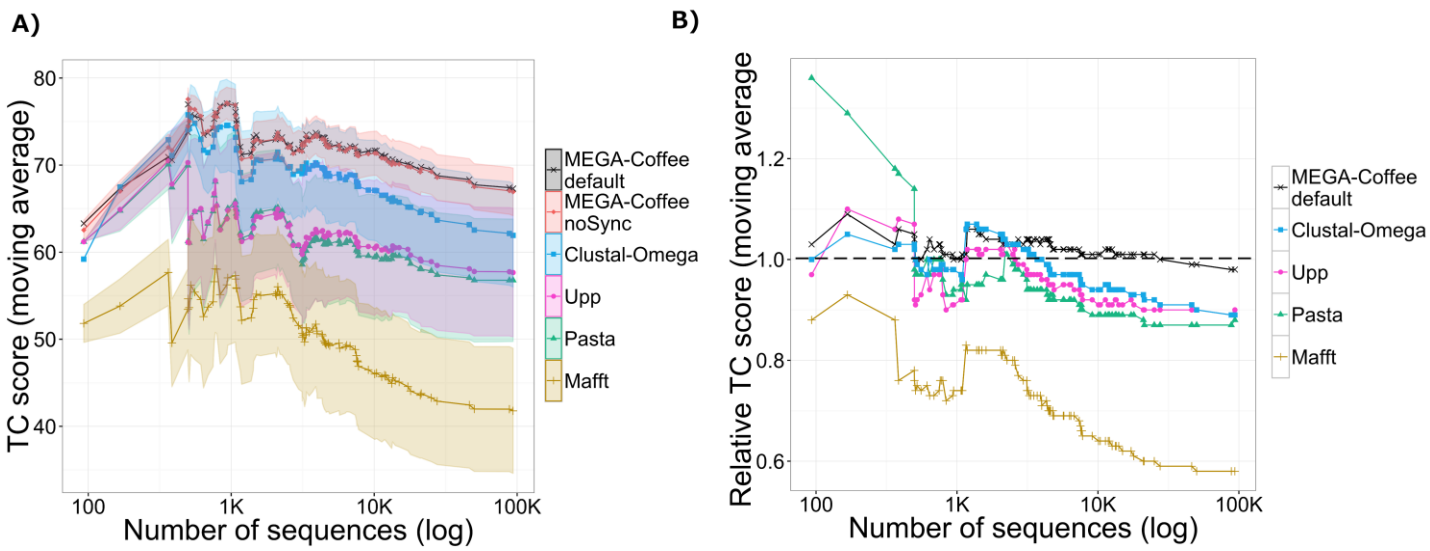


Figure 1. Benchmarking MEGA-Coffee on HOMFAM benchmark dataset using TC score. (A) MEGA-Coffee outperforms existing large-scale aligners in terms of accuracy. Moreover MEGA-Coffee default has a zero stdev of accuracy, suggesting it will always deliver the same alignment regardless of the sequence input-order. This is not the case for the other aligners, for which the delivered alignment accuracy depends on sequence-input order. **(B)** When relative accuracies are compared we see that MEGA-Coffee accuracy degrades less with the numbers of sequences, compared to the other methods. This is especially true for large datasets (>10000 seqs). Accuracy and Stdev of accuracy for MEGA-Coffee, Clustal-Omega, Mafft, Pasta and Upp with respect to the number of sequences. Points represent cumulative moving averages of TC scores; x-axis, logarithmic and y-axis, linear scale and correspond to a family in HOMFAM Benchmark dataset. The error-area represents the stdev of accuracy.

Figure 1 and Sup. Fig 1 shows the accuracy results of MEGA-Coffee, Clustal-Omega, Mafft, Pasta and Upp on every test case in HOMFAM. The graph is plotted as moving average TC score for all test cases with N or fewer numbers of sequences. N is plotted on the horizontal axis using log scale. From figure 1A two basic conclusions can be drawn: first that MEGA-Coffee outperforms in terms of MSA accuracy the existing large-scale aligners; second that MEGA-Coffee is less affected by alignment uncertainty compared to the other methods. This makes MEGA-Coffee a more stable aligner (with the default version of MEGA-Coffee delivering always the same MSA (ACC STEDV = 0)). These conclusions prove especially true when the aligners are asked to align medium (3000<sequences<10000) and large datasets (>10000 sequences). While on small datasets (<3000 sequences) MEGA-Coffee achieves slightly better results than the second best performing on average aligner Clustal-Omega, 72.16% and 69.32% respectively (Table 1), on medium and large size datasets it reaches 70.06% and 51.29% respectively, meaning almost 6 and 9 points of accuracy higher than Clustal-Omega (64.41% and 42.69% respectively) (Table 1). Here we have to note that on large-size datasets Upp and Pasta deliver MSAs of higher quality compared to Clustal-Omega (46.60%, 46.44% and 42.69% respectively) but not MEGA-Coffee (51.29%). When

it comes to the effect of alignment uncertainty, that may cause aligners to deliver MSAs of different quality whenever the same sequences are provided in different input-orders, MEGA-Coffee (noSync behavior) is less affected (average STDEV of accuracy 2.74%) than Clustal-Omega – 5.94%, Pasta – 7.03%, Upp – 7.38% and Mafft – 7.18% (Table 1). MEGA-Coffee default behavior is to always deliver the same high quality MSA regardless of sequence input-order. It achieves this by performing initially an alphanumeric ordering of the sequences and then putting in the left node of the guide-tree the longest sequence or profile MSA. When MEGA-Coffee is run with the "--no-sync" flag both this actions are turned off.

Table 2. HOMFAM benchmark results for Clustal-Omega when provided a MEGA-Coffee tree. Accuracy achieved in Total Column scores (TC) and Sum of Pairs (SoP) is shown for different family size ranges. The average run time in minutes is also shown for every range.

	OVERALL 93 < N < 93,681 (94 families)			**SMALL** 93< N < 2,957 (41 families)			**MEDIUM** 3,127 < N < 9,105 (33 families)			**BIG** 10,099 < N < 93,681 (20 families)		
Aligner	TC (%)	SoP (%)	Time (m)	TC (%)	SoP (%)	Time (m)	TC (%)	SoP (%)	Time (m)	TC (%)	SoP (%)	Time (m)
MEGA-Coffee	66.98	84.21	**4**	72.16	87.19	**0.5**	70.06	86.50	**4**	51.29	74.35	**12**
Clustal-Omega	61.93	80.64	39	69.32	85.73	16	64.41	82.67	40	42.69	66.87	86
Clustal-Omega with MEGA-Coffee tree	**67.63**	**84.53**	29	71.13	86.80	10	**72.34**	**87.82**	34	**52.65**	**74.47**	65
Clustal-Omega iter 2	64.77	81.61	166	71.94	86.89	81	68.35	84.62	188	44.15	65.80	304
Clustal-Omega iter 2 with MEGA-Coffee tree	66.28	82.73	160	**73.77**	**87.81**	87	69.16	85.31	162	46.17	68.04	304

Another trend that the figure reveals is that alignment accuracy seems to be degrading with larger numbers of sequences. To investigate this further we asked whether the difficulty in aligning the families in HOMFAM is due to the big number of sequences or if it reflects the "seed" sequences (reference sequences with known structure) for these families being harder to align. To answer this question we calculated (Fig. 1B, Sup. Fig. 1B) the ratio between the accuracy of reference sequences when aligned with the rest of the sequences and the accuracy one would obtain if aligning the reference sequences alone (relative accuracy). If this perspective ratio equals 1, it means that the reference sequences are aligned in the same way when aligned alone or when aligned along with the rest sequence members of the family. Otherwise a ratio lower than 1

indicates that the MSA method produces MSAs of lower quality than one would obtain if aligning the seed sequences alone a ratio is higher than 1 it indicates the opposite. As demonstrated in Fig. 1B MEGA-Coffee, Clustal-Omega, Pasta and Upp seem to be improving the alignment of the reference sequences (ratio>1) when adding up to 1000 sequences, passed that point the improvement over the direct seed alignments becomes limited (ratio=1) and starts degrading above 5000 sequences. MEGA-Coffee appears to be less affected than the other methods. When considering the very datasets (>10000 seqs) the relative improvement of MEGA-Coffee over the other methods is especially significant. Figures 1B and Sup fig. 1B illustrate how sensitive the algorithmic framework behind every method is, when it comes to the number of sequences they are asked to align. In an ideal world alignment methods should not care about the number of sequences to align (given that the extra sequences bring more information than noise). This is not the case though for existing methods. Although MEGA-Coffee, also sees its accuracy degrading with higher number of sequences, it is less affected by it compared to other methods, resulting in higher performance with bigger numbers of sequences and making MEGA-Coffee algorithmic framework a more prominent framework to tackle this problem. This is thus a future direction we are working on.

MEGA-Coffee owes it's performance to the StarClust algorithm, (Supplementary Material). We show here that this improved clustering is the driving force behind the improved performances of MEGA-Coffee. We did show this by measuring the effect of feeding ClustalOmega with the StarClust guide trees (Figure 2). The results are very clear and indicate that when Clustal-Omega is fed with the MEGA-Coffee guide-trees, it generates alignments of accuracy comparable to MEGA-Coffee and significantly better than its own default behavior. Because StarClust runs much faster than mBed, this improvement comnes along with faster computation by ClustalO. When compared against Clustal-Omega run with 2 iterations (Table 2), Clustal-Omega with MEGA-Coffee tree is still much more accurate. It is worth mentioning that when Clustal-Omega is asked to iterate over the MSA generated from the MEGA-Coffee tree, Clustal-Omega degrades the accuracy of the MSA. This experiment highlights that MEGA-Coffee guide-trees, generated using StarCust, are superior to Clustal-Omega guide-trees, generated using mBed (Table 2).

Another angle to address the problem of alignment uncertainty is to identify and remove the sequences that will tend to be noisier and force the rest of the sequences to align in peculiar ways. This approach was used to generate very large high quality alignments used for evolutionary analysis purposes [26]. MEGA-Coffee offers this possibility though its trimming algorithm. Table 3 and Sup. Table 3 show that on average when sequences in HOMFAM are trimmed, MEGA-Coffee is able to deliver MSAs of significantly increased accuracy and stability, when compared to random removal of sequences. Trimming, though, does not have the same effect on small and large

datasets. When applied to small datasets (<3000 sequences) it has little to no effect in both terms of accuracy and stability, but when applied to large datasets (>10000 sequences) MEGA-Coffee's performance increases from 51.30% to 57.23% at 25% trimming, 57.85% at 50% trimming and 62.36% at 75% trimming. At the same time instability decreases with the STDEV of accuracy dropping from 3.48% to 2.86% at 25% trimming, 3.02% at 50% trimming and 1.70% at 75% trimming. Although an increase in both accuracy and stability can be observed when we just randomly remove the same amount of sequences, due decrease in sequence number, it is not as significant as when sequences are removed using MEGA-Coffee's trimming algorithm.

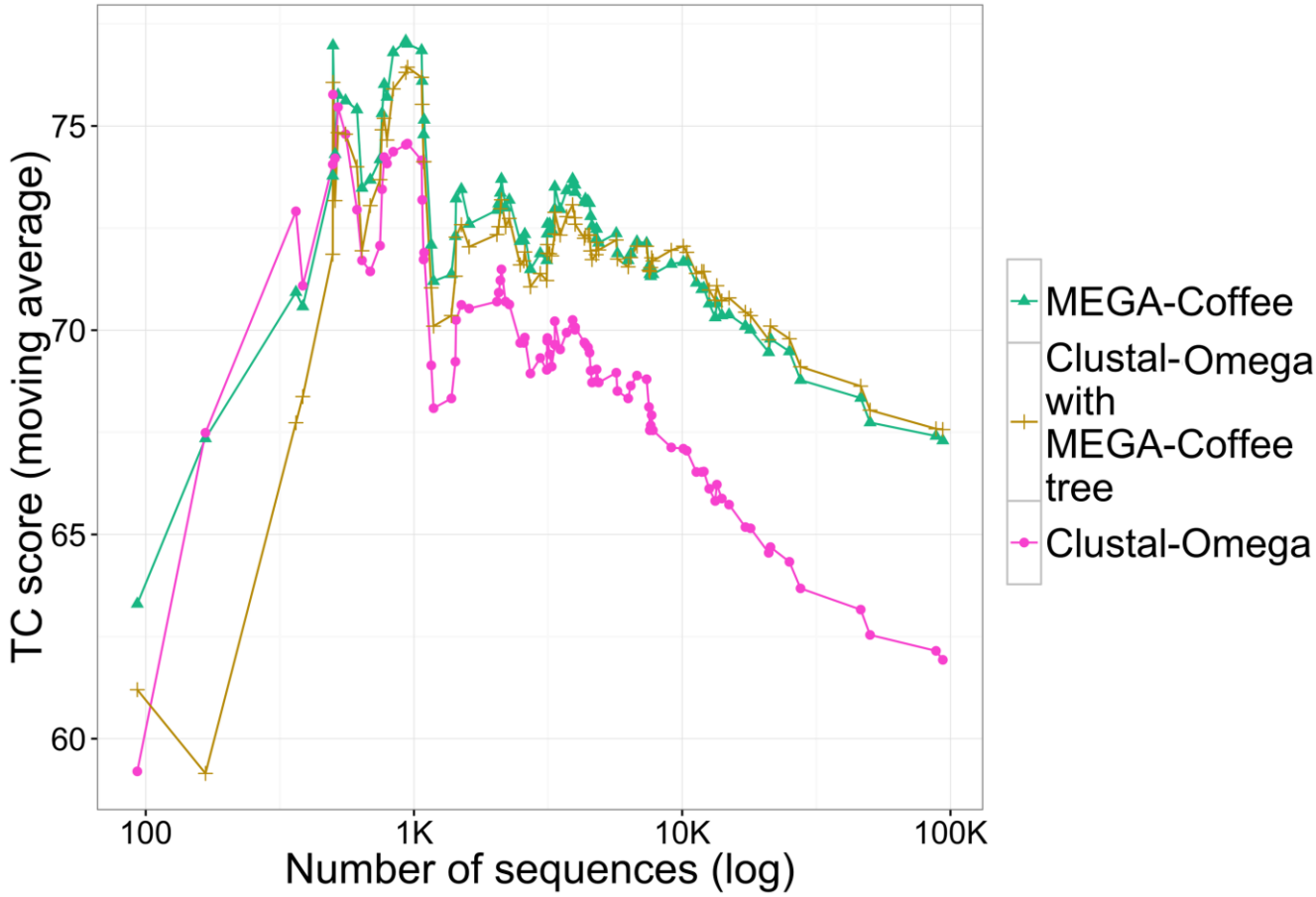


Figure 2. Clustal-Omega performance when given a MEGA-Coffee guide-tree in comparison with MEGA-Coffee and Clustal-Omega default performance. Clustalo-Omega achieves superior performance when using the MEGA-Coffee guide-tree to construct an MSA. This highlights the critical role guide-tree plays and indicates that MEGA-Coffee delivers better guide-trees, resulting thus in high quality MSAs. Accuracy for MEGA-Coffee, Clustal-Omega with MEGA-Coffee tree and Clustal-Omega default, with respect to the number of sequences. Points represent cumulative moving averages of TC scores; x-axis, logarithmic and y-axis, linear scale and correspond to a family in HOMFAM Benchmark dataset.

When it comes to very small datasets (<100 sequences) (Sup. Table 1 and 2) MEGA-Coffee achieves comparable performance to the best methods, because of the use of consistency. When looking at run times Mafft default is exceptionally fast in the smaller test cases (Sup. Table 1 and 2) and Mafft parttree is very fast on the bigger families (Table 1 and Sup. 3). In the case of the bigger families in HOMFAM benchmark MEGA-Coffee is 4x slower overall than Mafft, but 10x faster than Clustal-Omega and Pasta, and 5x times faster than Upp.

Table 3. MEGA-Coffee HOMFAM benchmark results on datasets after normal trimming and random sequence removal (trim rand). Accuracy achieved and its STDEV in Total Column scores (TC) are shown for different family size ranges. The average run time in minutes is also shown for every range.

	OVERALL 93 < N < 93,681 (94 families)			**SMALL** 93< N < 2,957 (41 families)			**MEDIUM** 3,127 < N < 9,105 (33 families)			**BIG** 10,099 < N < 93,681 (20 families)		
Trimming	TC (%)	STDEV (%)	Time (m)	TC (%)	STDEV (%)	Time (m)	TC (%)	STDEV (%)	Time (m)	TC (%)	STDEV (%)	Time (m)
trim 75	**71.78**	**1.47**	5	72.66	1.75	0.5	76.38	**0.97**	4	**62.36**	**1.70**	14
trim 50	71.65	1.85	5	**74.50**	**1.73**	0.5	**76.49**	1.30	4	57.85	3.02	16
trim 25	70.95	1.97	6	74.18	1.90	0.5	75.27	1.53	4	57.23	2.86	19
trim rand 75	70.18	1.96	5	73.75	1.62	0.5	74.12	2.05	4	56.34	2.52	14
trim rand 50	68.70	2.48	5	72.83	2.00	0.5	72.83	2.69	4	53.43	3.11	16
trim rand 25	67.44	2.85	6	71.79	2.58	0.5	71.44	2.68	4	51.95	3.68	19

3.4 Discusion

Since the mid 1980s the main breakthroughs in MSA methods have been progressive alignment and the use of consistency. Most recent work has concerned the development of “fast” guide tree construction algorithms (mBed used by Clustal-Omega, Mafft PartTree) so as to speed up alignment estimation. Both PartTree and mBed are fast but at the expense of accuracy, as judged by the benchmarks. Thus this new generation of aligners has been focusing mainly in making alignment methods able to scale with big number of sequences while trying to maximize MSA accuracy. Few attempts however were done to propose solutions to the problem of alignment uncertainty, which results in highly unstable MSAs the larger the evolutionary distance and the number of sequences to be aligned is. This instability can so easily occur by just changing the sequence-input order, which in turn raises many concerns about the confidence of the MSA

models and even the reproducibility of the MSA itself and every downstream procedure based on it.

We demonstrated that the major cause of MSA instability is the unordered nature of child nodes in the binary guide tree. Indeed, while the dynamic programming (DP) algorithm used to do pairwise alignments at every node is guaranteed to return the best possible scoring alignment of the two considered child nodes, this algorithm may deliver different – but equally optimal – pairwise alignments when inverting the input order of the sequences. This effect - sometimes referred to as high-road/low-road resolution - results from the handling of tiebreaks in the DP implementation and is an issue when progressively assembling MSAs. Therefore, it should come as no surprise that the larger a dataset and the lower its sequence identity level, the harder and the less stable its alignment becomes. This explains why very large datasets are so sensitive to the re-ordering effect.

MEGA-Coffee is a novel and highly accurate method for producing stable and biological meaningful alignments of any number of sequences. We argue that the MEGA-Coffee, accuracy and scaling to higher number of sequences are essential for coping with the ever-increasing information of biological databases and for better problem-driven data analyses, ultimately leading to higher-quality biological insights. Aside from its improved computational efficiency, MEGA-Coffee, offers dramatic improvements over the standard T-Coffee and the majority of existing alignment methods on large and/or difficult-to-align data sets in both terms of speed and accuracy. At the same time MEGA-Coffee is less affected by alignment uncertainty compared to existing methods. This makes it a more stable aligner that guarantees delivery of the same high quality alignments, regardless of sequence input-order, ensuring thus the reproducibility of data and biological interpretations.

The key to increasing and/or retaining high accuracy with a higher number of sequences, while minimizing the effects of alignment uncertainty, is the use of our novel clustering method, StarClust. The notion behind this algorithm design is that even the best alignment methods have difficulty aligning sequence data sets when they are large and/or have evolved with many substitutions and indels. The MEGA-Coffee decomposition technique seeks to break the large data set into smaller subsets, so that each of the smaller subsets has higher sequence identity and thus smaller evolutionary distance, making their alignment much easier and less erroneous. Moreover, the merger technique does not undo the alignments of the subsets but seeks to merge the improved subset alignments into a larger MSA, which maintains the improved accuracy. In general, this means that very large data sets that cannot be analyzed by standard alignment methods in the

same timeframes (eg. Standard T-Coffee, Probcons, MSAprobs etc.), can be analyzed using MEGA-Coffee.

Another way of dealing with the problem of alignment uncertainty is by trying to identify and remove the sequences that will tend to be noisier and force the rest of the sequences to align in peculiar ways. MEGA-Coffee offers this possibility thanks to its trimming algorithm. We demonstrate here that the MEGA-Coffee trimming approach is able to deliver MSAs of significantly increased accuracy and stability. Trimming can be a beneficial approach, when it comes to improving the alignment quality for large datasets, for which we can afford to remove a significant number of sequences without affecting much the information the MSA carries. Though, we do not expect trimming to benefit the same large and small datasets. In the case of small datasets (<1000), removing sequences can prove more crucial and lead to removal of more information than noise.

3.5 Supplementary Material

3.5.1 Supplementary Tables

Supplementary table 1. Balibase results. Total column scores (TC) are shown for different families in Balibase that correspond to different identity ranges; the second column is the average score over all test cases. The average run time in seconds is shown in the last column along with whether the method is consistency based or not.

Aligner	**Overall** TC (%) (218 families)	**BB11** TC (%) (38 families)	**BB12** TC (%) (44 families)	**BB2** TC (%) (41 families)	**BB3** TC (%) (30 families)	**BB4** TC (%) (49 families)	**BB5** TC (%) (16 families)	**Overall AVG Time** (sec) **/ Consistency**
MSAprobs	**60.7**	44.1	86.5	46.4	60.7	62.2	60.8	57 / Yes
Probalign	**58.9**	45.3	86.2	43.9	56.6	60.3	54.9	46 / Yes
Mafft (auto)	**58.8**	43.9	83.1	45.0	58.1	60.5	59.1	7 / Mostly (203/218)
Probcons	**55.8**	41.7	85.5	40.6	54.4	532	57.3	60 / Yes
MEGA-Coffee	**55.5**	40.4	85.7	43.8	56.4	50	56.3	50 / Yes
ClustalO	**55.4**	35.8	78.9	45.0	57.5	57.9	53.3	3 / No

Supplementary table 2. Prefab results. Total column scores (TC) are shown for different percent identity ranges; the second column is the average score over all test cases. The average run time in seconds is shown in the last column along with whether the method is consistency based or not.

Aligner	**Overall** TC (%) (1682 families)	**0<ID<20** TC (%) (912 families)	**20<ID<40** TC (%) (563 families)	**40<ID<70** TC (%) (117 families)	**70<ID<100** TC (%) (90 families)	**Overall AVG Time** (sec) **/ Consistency**
MSAprobs	**73.7**	59.1	88.9	96.5	97.1	30 / Yes
Mafft (auto)	**72.1**	56.9	87.6	96.1	97.9	3 / Yes
MEGA-Coffee	**71.9**	56.8	87.3	96.3	97.7	29 / Yes
Probalign	**71.9**	56.3	88.1	96.1	97.7	21 / Yes
Probcons	**71.7**	56.2	87.6	95.5	97.2	28 / Yes
ClustalO	**70**	53.5	86.6	96.7	98	1 / No

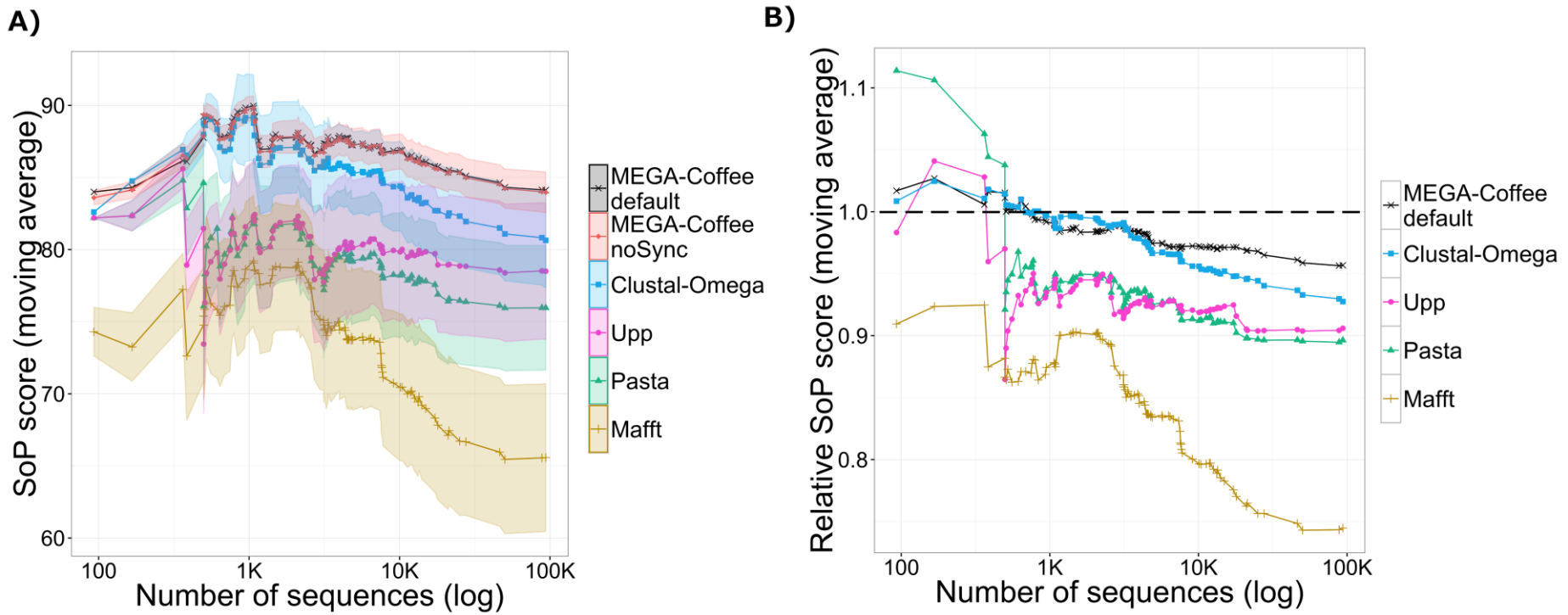
Supplementary table 3. HOMFAM results. HOMFAM benchmark results for different large-scale aligners and the MEGA-Coffee benchmark results on these datasets after normal trimming and random sequence removal (trim rand) Accuracy achieved and its STDEV in Total Column scores (TC) and Sum of Pairs scores (SoP), as well as their STDEV are shown for different family size ranges. The average run time in minutes is also shown for every range.

		OVERALL 93 < N < 93681 (94 families)					**SMALL** 93< N < 2957 (41 families)					**MEDIUM** 3127 < N < 9105 (33 families)					**BIG** 10 099 < N < 93 681 (20 families)				
Aligner		TC (%)	TC STDEV (%)	SoP (%)	SoP STDEV (%)	Time (m)	TC (%)	TC STDEV (%)	SoP (%)	SoP STDEV (%)	Time (m)	TC (%)	TC STDEV (%)	SoP (%)	SoP STDEV (%)	Time (m)	TC (%)	TC STDEV (%)	SoP (%)	SoP STDEV (%)	Time (m)
MEGA-Coffee	**trim 75**	**71.78**	**1.47**	**87.37**	**0.83**	**5**	72.66	1.75	87.18	**0.92**	**0.5**	76.38	0.97	90.70	0.53	4	62.36	1.70	82.26	1.16	14
	trim 50	**71.65**	1.85	87.42	1.02	**5**	74.50	1.73	88.47	0.97	0.5	76.49	1.30	90.71	0.71	4	57.85	3.02	79.85	1.63	16
	trim 25	70.95	1.97	86.88	1.08	6	74.18	1.90	88.38	1.03	0.5	75.27	1.53	89.95	0.84	4	57.23	2.86	78.74	1.59	19
	trim rand 75	70.18	1.96	86.22	1.07	5	73.75	1.62	88.19	0.88	0.5	74.12	2.05	88.91	0.94	4	56.34	2.52	77.74	1.68	14
	trim rand 50	68.70	2.48	85.18	1.33	5	72.83	2.00	87.53	1.07	0.5	72.83	2.69	88.27	1.27	4	53.43	3.11	75.27	1.95	16
	trim rand 25	67.44	2.85	84.42	1.48	6	71.79	2.58	86.85	1.22	0.5	71.44	2.68	87.44	1.31	4	51.95	3.68	74.47	2.31	19
MEGA-C default		**66.98**	**0**	**84.21**	**0**	**4**	**72.16**	**0**	**87.19**	**0**	**0.5**	**70.06**	**0**	**86.50**	**0**	**4**	**51.29**	**0**	**74.35**	**0**	**12**
MEGA-C noSync		66.92	2.74	83.93	1.45	4	71.73	2.28	86.75	1.09	0.5	70.72	2.88	86.67	1.42	4	50.80	3.48	73.64	2.22	12
ClustalO		61.93	5.94	80.64	3.65	39	69.32	5.12	85.73	2.99	16	64.41	6.46	82.67	3.97	40	42.69	6.77	66.87	4.46	86
ClustalO iter 2		64.77	-	81.61	-	166	71.94	-	86.89	-	81	68.35	-	84.62	-	188	44.15	-	65.80	-	304
Upp		57.66	7.38	77.63	5.17	20	60.83	7.67	79.05	5.02	5	60.43	7.66	80.18	5.77	22	46.60	6.32	70.50	4.51	44
Pasta		56.78	7.03	75.10	4.89	41	60.70	7.41	78.57	4.81	7	58.19	6.74	75.50	4.97	38	46.44	6.74	67.30	4.91	113
Mafft		41.78	7.18	64.59	5.68	1	51.58	7.31	73.43	4.93	0.5	40.17	7.83	65.25	6.61	1	24.37	5.85	45.38	5.70	4

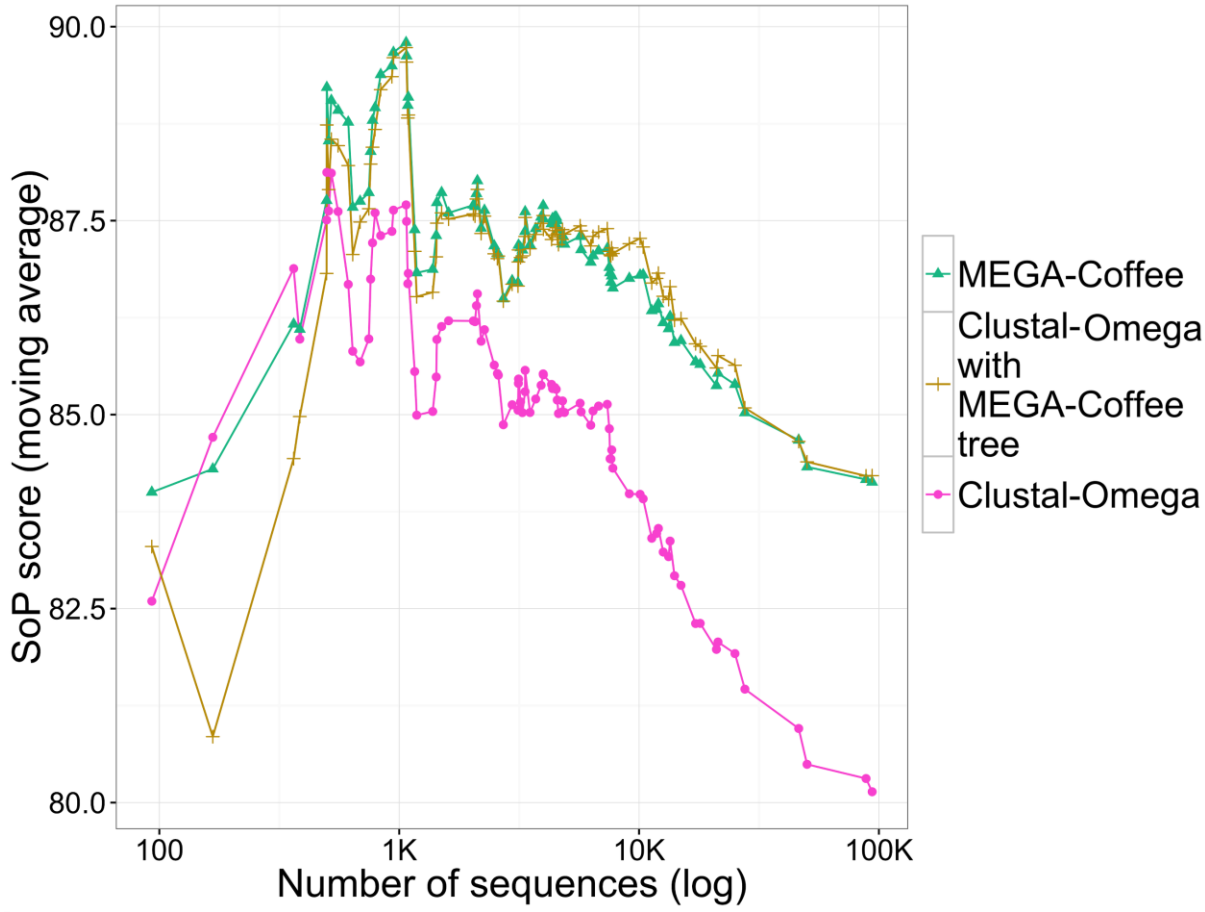
Supplementary table 4. MSA programs used to benchmark MEGA-Coffee against. The programs can be obtained from the URLs given below.

Tool	Url
T-Coffee Version 11	http://www.tcoffee.org/
Clustal-Omega, v1.2	http://www.clustal.org/omega/
Mafft 6.857	http://mafft.cbrc.jp/alignment/software/source.html
MSAProbs 0.9.4	http://sourceforge.net/projects/msaprobs/files/
Probalign v1.4	http://cs.njit.edu/usman/probalign/
Probcons version 1.12	http://probcons.stanford.edu/download.html

3.5.2 Supplementary Figures



Supplementary figure 1. Benchmarking MEGA-Coffee on HOMFAM benchmark dataset using SoP score. (A) MEGA-Coffee outperforms existing large-scale aligners in terms of accuracy. Moreover MEGA-Coffee default has a zero stdev of accuracy, suggesting it will always deliver the same alignment regardless of the sequence input-order. This is not the case for the other aligners, for which the delivered alignment accuracy is highly linked to sequence-input order. **(B)** When relative accuracies are compared we see that MEGA-Coffee accuracy degrades less with the numbers of sequences, compared to the other methods. This is especially true for large datasets (>10000 seqs). Accuracy and Stdev of accuracy for MEGA-Coffee, Clustal-Omega, Mafft, Pasta and Upp with respect to the number of sequences. Points represent cumulative moving averages of SoP scores; x-axis, logarithmic and y-axis, linear scale and correspond to a family in HOMFAM Benchmark dataset. The error-area represents the stdev of accuracy.



Supplementary figure 2. Clustal-Omega performance when given a MEGA-Coffee guide-tree in comparison with MEGA-Coffee and Clustal-Omega default performance. Clustalo-Omega achieves superior performance when using the MEGA-Coffee guide-tree to construct an MSA. This highlights the critical role a good guide-tree plays and indicates that MEGA-Coffee delivers better guide-trees, resulting thus in high quality MSAs. Accuracy for MEGA-Coffee, Clustal-Omega with MEGA-Coffee tree and Clustal-Omega default, with respect to the number of sequences. Points represent cumulative moving averages of SoP scores; x-axis, logarithmic and y-axis, linear scale and correspond to a family in HOMFAM Benchmark dataset.

3.5.3 Materials and Methods

3.5.3.1 MEGA-Coffee Algorithm

The MEGA-Coffee algorithm consists of three major steps: (I) filtering out the sequences that have up to 95% sequence identity, (II) clustering the remaining sequences utilizing a hierarchical greedy clustering algorithm, which we call “StarClust”, (III) aligning the clustered sequences. After the final alignment has been generated, thanks to its trimming algorithm, MEGA-Coffee can detect and remove noisy and/or problematic sequences and realign only the remaining ones.

3.5.3.1.1 Filtering

The filtering algorithm starts by first translating each sequence into a corresponding one that consists of a 6 amino acid alphabet instead of 20 (the following reduced alphabet was used for this *("AST =>A, CP =>C, DEHKNQR =>D, FWY=>F, G, ILMV=>I")).* This transformation allows one to quickly group not only identical, but also very similar sequences, by means of efficient hashing. Then the remaining sequences are compared to each other by estimating the hamming distance, which has a complexity of O(n). If two given sequences have a hamming distance higher than 90% then they are grouped together. Sequences that are grouped by this method have a sequence identity that ranges from 100 to 90%. Finally the grouped sequences are removed from the dataset to be aligned, allowing in it only one representative sequence for each group, and are introduced into the final alignment based on their representatives.

3.5.3.1.2 Clustering

The StarClust algorithm is a hierarchical bottom-up greedy approach that divides the sequences to be aligned into a hierarchy of clusters. A cluster is defined by one sequence, known as the centroid-representative sequence. Every sequence in the cluster (member sequence) must have similarity above a given identity and coverage threshold T with the centroid, so that two criteria are fulfilled: (I) all centroids have similarity < T to each other, and (II) all member sequences have similarity >= T to a given centroid sequence.

StarClust will perform a maximum of 8 rounds of clustering. In the first round each sequence is assigned to a cluster if the identity and coverage from the centroid-representative sequence is >=80%. In the following sub-clustering rounds the identity and coverage threshold (T) is lowered by 10%, each time, until it reaches T=10%, which is the lowest identity and coverage threshold mega can handle. Sub-clustering of the sequences can stop if after a clustering round a specific number of clusters is reached (the default threshold is 5000 clusters for a good speed to sensitivity tradeoff and can be changed with the parameter --cluster_number). Once a round of clustering is completed the sequences within the clusters are progressively aligned following the order of an all-against-all Smith-Waterman distance-based guide-tree. Next each centroid-representative sequence is scanned to check if gaps have been induced to it due to aligning, if yes then the gaps are filled with parts of other sequences and the representative is transformed to a hybrid sequence, the “star-sequence”. The name originates from the concept of center star alignment, since these special sequences are the center stars. Finally these star-sequences are passed to the next round of clustering.

StarClust clusters the sequences by using the same greedy incremental algorithm as implemented by Holm and Sander [155]. It starts by sorting the sequences in order of decreasing length, so that the longest sequence appears first. This sequence becomes the centroid-representative sequence of the first cluster. Then each remaining sequence is compared to the existing centroid-representative sequences and if identity and coverage are above a given threshold (T) it is assigned to that cluster; otherwise it becomes the centroid of a new cluster.

Identity and coverage are determined by computing Smith-Waterman alignments between the actual sequences and the centroid-representative ones. To speedup the SW alignment computation, which has a complexity of O(n*m), we use a really fast vectorized implementation of it [156] [157].

3.5.3.1.3 Aligning

After clustering is completed, the alignment of the sequences left follows. Initially all-against-all Smith-Waterman distances are calculated. Then these distances are turned into a guide-tree using UPGMA. Next the alignment starts; following the tree topology from leafs to root. For the estimation of profile-profile alignments HHalign [158] is used. Once this core alignment is estimated, the clustered aligned sequences are introduced to it using the star-sequence as an anchor point and the transitivity rule. Finally the filtered out sequences are also put back into it and the final alignment is outputted.

3.5.3.1.4 Trimming

In an effort to increase MSA accuracy and stability of large datasets we used a trimming procedure to identify and remove noisy sequences that when present lead to decrease of the MSA quality. Our trimming procedure is a MSA post-processing methodology that reads and renders protein or nucleotide alignments. It starts by reading all rows in an alignment and computes a score (Sr) for each gap in them. This score is computed by assigning a penalty of 1 for every first gap that follows after a letter, dividing it by the number of consecutive gaps that follows this gap (if any gaps follow). Then the score for each column is computed (Sc) and it corresponds to the fraction of sequences without a gap in the particular column (number of gaps expressed in (Sr), divided by number of letters in the particular column). A final "gap responsibility" score (Sg) is then calculated for each sequence by adding up the (Sc) score of every non-gap column of that sequence. The sequences with the highest "gap responsibility" are chosen and then trimmed, in order to remove sequences introducing the most fuzzy gaps, according to the trimming threshold set by the user. The remaining sequences are then re-aligned.

To evaluate the performance of the trimming procedure we compared it against randomly removing sequences (Table 1, Sup. Table 3). In this paper we call *"trimming"* the trimming performed by our trimming procedure, as described above, and *"random trimming"* the random sequence removal.

3.5.3.1.5 Implementation and Availability

This new algorithm has been implemented from scratch using C++. Different parts have been parallelized using OpenMP. MEGA-Coffee is licensed under the GNU Lesser General Public License. Source code is available at https://bitbucket.org/mariach/mega-coffee. MEGA-Coffee is available as a command line program, which uses GNU-style command line options, making it easy to integrate into existing pipelines.

This new implementation is part of the T-Coffee package. The initial T-Coffee consistency code, which was written in C, was translated into C++ and became part of MEGA-Coffee. We use the OpenMP library to enable multithreaded computation of pairwise distances and alignment match states.

3.5.3.2 Benchmark procedure

In this paper, we present results from a range of packages tested on three common benchmarks: BAliBASE 3 [159], Prefab 4.0 [108] and of HOMFAM [14]. For these tests, we report results using the default settings for all programs, with one exception; needed to allow Mafft [13] to align the biggest test cases in HOMFAM. Mafft consists of a series of programs that can be run separately or called automatically from a script with the "-auto" flag set. This flag chooses to run a slow, consistency-based program (L-INS-i) when the number and length of sequences is small. When the number exceeds inbuilt thresholds, a conventional progressive aligner is used (FFT-NS-2). The latter is also the program that is run by default if Mafft is called with no flags set. For very large data sets, the "-parttree" flag must be set on the command line and a fast guide tree calculation is then used.

The results for the BAliBASE benchmark tests are shown in Sup. Table 1. BAliBASE is divided into six reference sets with different properties. Average scores are given for each reference, along with average run times and average total column (TC) scores, that give the proportion of the total alignment columns in the reference alignment that have been recovered in the test alignment. Bali-score [159] was used to estimate the TC score. A score of 100.0 indicates perfect agreement with the benchmark. In 203 out of 218 BAliBASE test cases, the number of sequences is small, so the Mafft (auto) runs L-INS-i, which is the slower, more accurate program that uses the consistency heuristic [160]. This is also used by MSAprobs [23], Probalign [161] and Probcons [10]. These programs are all restricted to small numbers of sequences but tend to give the most accurate alignments. This is clearly reflected in the times and average scores in Table 1. The times range from 3 seconds up to 1 minute for these packages and the MSA accuracy ranges from 55% to 61% of correct columns. Clustal-Omega, which does not utilizes consistency takes 3 sec for the same runs and has an accuracy level that is lower than that of MSAprobs and T-Coffee. Other programs that use progressive alignment were not included in our benchmark, as they have already been reported [14] to show a considerable drop in accuracy when compared to the consistency-based programs and Clustal-Omega.

The Prefab benchmark [12] test results are shown in Sup. Table 2. Total Column Scores were estimated using the qscore from the Prefab package. Here, the results are divided into five groups according to the percent identity of the sequences. The overall scores range from 53 to 73% of columns correct. The consistency-based programs MSAprobs, Mafft L-INS-i, T-Coffee, Probalign, and Probcons are again the most accurate but have long run times. Clustal-Omega is faster but

less accurate compared to consistency-based methods. The times range from 1 second up to 30 seconds.

All the programs that were used for benchmarking are reported in Sup. Table 4 and can be obtained from the URLs given in the table.

4. Improved numerical stability of in silico -omics analyses across clouds and clusters

Di Tommaso P*, Chatzou M*, Floden E*, Palumbo E, Prieto-Barja P, Notredame C (2016). Improved numerical stability of in silico -omics analyses across clouds and clusters. *Nature Biotechnology, Under Review*

*joint first authors

4.1 Abstract

Reproducing routine bioinformatics analysis is challenging owing to a combination of factors hard to control for. Nextflow is a flow management framework that uses container technology to insure efficient deployment and reproducibility of computational analysis pipelines. Third party pipelines can be ported into nextflow with minimum re-coding. We used RNA-Seq quantification and phylogeny reconstruction examples to show how two seemingly irreproducible analyzes can be made stable across platforms when ported into Nextflow.

4.2 Main Text

The increasingly worrying issue of irreproducibility in experimental biology is a natural consequence of -omics analysis relying on readout methods of growing complexity [162][163][164]. While a common definition is still being intensely debated [165] [166], it is generally agreed that the most obvious source of computational irreproducibility results from inadequate good practices when deploying software and databases [167][168]. Beside poor standards [169], computational irreproducibility is also the consequence of less apparent factors such as numerical instability arising from low-level variations across computational platforms [170]. This issue is especially relevant when using high-performance computational (HPC) environments, the type on which -omics analyses are routinely carried out [171]. Here we report on the significance of such fluctuations when repeating the same *in silico* procedure on different operating system configurations. This instability is especially relevant at a time when precision medicine is imposing an unprecedented burden on the replicability of data analyses. We show that novel virtualization methods bring about a long-awaited solution to this issue and should play an important role in the current shift towards reproducible biology. We describe a solution based on our new workflow management tool called Nextflow, which uses the Docker technology for the multi-scale handling of containerized computation. Thanks to the proper deployment of these techniques, our results establish that it is possible to guarantee numerical stability across the more common UNIX-like environments.

In silico workflow management systems have recently become an integral part of biological analysis. These systems make it possible to rapidly prototype and deploy pipelines combining complementary software packages. In genomics, the simplest pipelines, such as Kallisto and Sleuth [172], involve the combination of an RNA-seq quantification method with a differential expression module. Complexity, however, grows rapidly when attempting to cover all aspects of a specific task. For instance, the Sanger Companion pipeline [173] bundles 39 independent software tools and libraries into a genome annotation suite. Handling such a large number of software packages along with their potentially incompatible dependencies is a challenge of its own. This issue is further worsened by the conflicting requirements to accommodate frequent software updates whilst maintaining the reproducibility of original results. High-throughput deployment of complex pipelines can also be problematic, owing to the very large number of intermediate files produced by the individual components. At this scale, likely hardware fluctuations combined with poor error handling can also lead to readout instability.

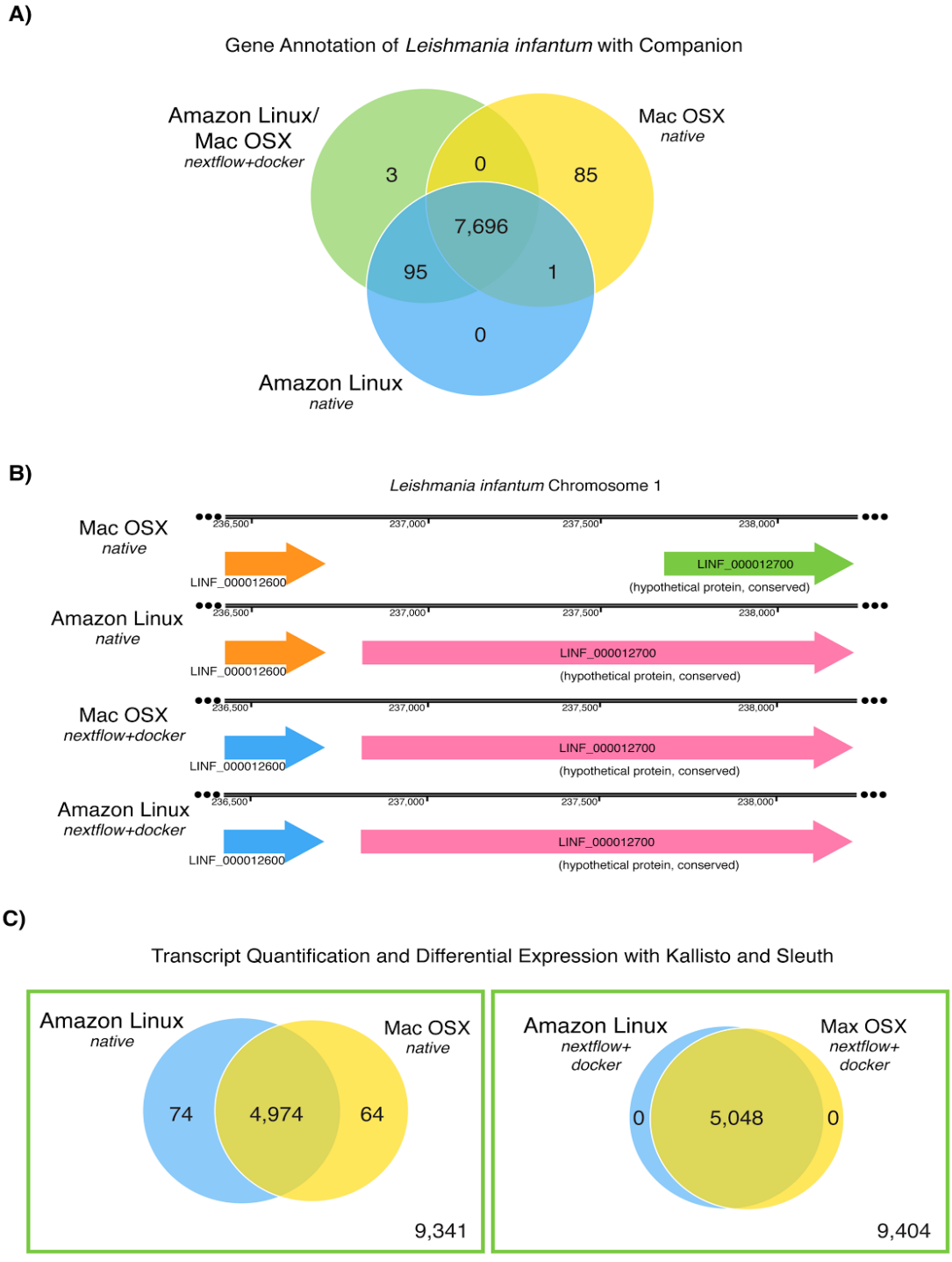


Figure 1. Figure 1: Nextflow produces stable analysis across different platforms. (a) *Leishmania infantum* clone JPCM5 genome annotation was predicted using a native and a dockerized (Debian Linux) version of the Companion eukaryotic annotation pipeline. The native and dockerized versions were run on Mac OSX and Amazon Linux platforms. The Venn diagram shows the existence of small, but significant discrepancies when comparing the genomic coordinates of predicted coding genes and non-coding RNAs, (**b)** some of these disparities including entire genes. Results were deterministic on each platform, and totally identical readouts were measured when deploying the dockerized version. **(c)** A similar comparison carried out on a Kallisto/Sleuth pipeline when looking for differentially expressed genes (q-value <0.01) in an RNA-seq experiment collected from human lung fibroblasts reveals a comparable fluctuation between the Mac OSX and the Amazon Linux platform. Both platforms produce are deterministic identical readouts when deploying the dockerized version of the pipeline (10.5281/zenodo.159153)

Nextflow has been designed to address numerical instability, efficient parallel deployment, error tolerance, execution provenance, and maintenance traceability. It is a domain-specific language (DSL) that enables rapid pipeline prototyping as well as the adaptation of existing pipelines written in any scripting language. A qualitative comparison between Nextflow and several related tools (Supplementary Table 1) illustrates well its unique combination of features, especially with respect to Bpipe, whose computing paradigm is probably the most similar. The most notable shortcoming of Bpipe is its lacks of support for multi-scale containerization, one of the features now supported by the newest generation of workflow managers, including Nextflow. We found this mode of containerization, which makes it possible to bundle entire pipelines, subcomponents and individual tools into their own containers, to be essential for numerical stability. Containers can be produced ad hoc by the authors or by following recently proposed standards (BioBoxes [174], Bioshadock [175] and AlgoRun [176]). Another key specificity of Nextflow is its full integration with software repositories like GitHub and BitBucket, as well as with cloud native support. Some of the most practical implications of this integration are highly relevant to computational reproducibility and the specific impact of GitHub was recently highlighted as one of the driving force behind the current data sharing effort [177]. Given a published and properly deposited analysis, this integration allows users to run any current or previous version of a pipeline, with GitHub integration ensuring the deployment of the correct version, containerization ensuring numerical stability, and cloud deployment allowing for rapid computation and effective scaling. When using this procedure, any set of results – a table, a graph, or quantities - can be associated with a single command line and referenced, updated, reproduced or improved on demand. In the last section of this manuscript we provide three concrete examples of this procedure.

Nextflow uses a functional reactive programming (FRP) model in which each operation is isolated in its own execution context. Its outputs are streamed to other operations through dedicated communication channels in a process similar to UNIX pipes. When doing so, parallelization is an implicit consequence of the way the input/output of each process is declared to be channelled into other processes. This approach spares users the need to implement an explicit parallelization strategy. Another advantage of Nextflow is its reliance on the dataflow programming paradigm, where tasks are started automatically once data is received through their input channels. The dataflow model is superior to alternative solutions based on a Make-like approach, such as Snakemake [178], in which computation involves the pre-estimation of all computational dependencies, starting from the expected results up until the input raw data. This procedure requires a directed acyclic graph (DAG), whose storage requirement is a limiting factor in very large computations. In contrast, as the top to bottom processing model used by Nextflow follows the natural flow of data analysis, it does not require a DAG. Instead, the graph it traverses is merely incidental and does not need to be pre-computed or even stored, thereby ensuring high scalability [179]. The use of communication channels between tasks also contributes to Nextflow´s

computational robustness, especially with respect to Snakemake, whose task executions sequence is defined by rules and patterns defined on the input/output file names. These dependencies make it difficult to deal with multiple/variable output files, and consequently often require the implementation of low level output management procedures to deal with a pipeline´s individual stages. In comparison, Nextflow can handle any data structure and output without being limited to files. Nextflow, along with the latest generation of workflow managers, is a relatively low level tool explicitly designed for bioinformaticians. This clearly sets it apart from Galaxy, one of the most popular workflow systems [180]. Galaxy adequately addresses the numerical stability issue thanks to a custom package manager called Tool Shed [181]. Yet, while its graphical user interface (GUI) offers very powerful support for *de novo* pipeline implementation by non-specialists, it also imposes a heavy development burden in the case of existing third party pipelines, especially when dealing with complex combinations of tools. This reimplementation requirement is not only unique to Galaxy, but also affects some of the newer tools such as Toil [182].

Nextflow in Action. In order to show the concrete effect of environmental variability on numerical stability, we used the Sanger Companion pipeline [173] to carry out a gene annotation prediction on the genome of *Leishmania infantum* (Methods). Although this compact eukaryotic genome should be a relatively easy target for such analysis, our results indicated variations across different UNIX platforms (Fig. 1a, b and Supp Table 2). This instability contrasts with the deterministic behavior measured on each individual platform. As the Companion pipeline has been largely implemented in Nextflow, we were able to confirm readout stability when deploying a dockerized version of this same pipeline across the three Unix-like operating systems. Gene annotation is not the only type of genomic analysis affected. We identified similar issues when using the Kallisto expression quantification tool combined with the Sleuth differential expression package [172]. In this case, variations in the identification of differentially expressed genes were observed when running the pipeline on two different systems (Fig. 1c). However, no such differences were observed when running a Nextflow dockerized version of the same pipeline. Finally, a comparable platform-dependent effect was also observed when estimating maximum likelihood trees with RaxML [183] (Supp. Fig. 3). These variations were effectively controlled for when deploying the dockerized version of the same pipeline. It is worth mentioning that all these computational experiments are available on GitHub (pipelines) and Zenodo (data/results) and are therefore entirely reproducible in their dockerized form (Methods).

Nextflow is a simple, yet powerful solution that effectively addresses the numerical instability issue when deploying pipelines across diverse computational platforms. Here we show that this instability, being very frequent, affects most types of modelling carried out *in silico*. Its overall impact on final readouts may appear modest, but the lack of effective solutions confronts users with a daunting challenge. Careful monitoring of database and software versions is simply not

enough. The lack of numerical stability can have severe consequences at all levels. When processing experimental data, it can compromise verifications and updates of previously established results. In a personalized medicine production environment it may result in treatment variations with potentially dramatic consequences. The rapidly growing Nextflow user community [184] [185] illustrates the pressing need for more robust computational frameworks. At a time when technology keeps evolving at a breathtaking pace, Nextflow offers a mature solution to one of the few technical issues that is likely to be long-lasting in both the academic and clinical setting: the control of numerical stability [186].

4.3 Author Contributions

C.N. directed the work, contributed ideas, and helped writing the manuscript, P.D.T. contributed ideas, implemented the Nextflow package and helped writing the manuscript, M.C. contributed ideas, helped implementing the Nextflow package, implemented NF pipelines and helped writing the ms. P.P.B. contributed ideas, helped implementing the Nextflow package, implemented NF pipelines and helped testing the software. E.P. contributed ideas and helped implementing the Nextflow package. E.F. contributed ideas, helped implementing NF pipelines and writing the manuscript.

4.4 Funding

This project was supported by the Center for Genomic Regulation and the Spanish Plan Nacional and the Spanish Ministry of Economy and Competitiveness, 'Centro de Excelencia Severo Ochoa 2013-2017'. M.C. and E.F. were supported by the La Caixa foundation.

4.5 Acknowledgements

We wish to thank Tony Ferrar for manuscript revision and helpful comments (http://theeditorsite.com), and Roderic Guigo for suggesting ENCODE applications.

4.6 Supplementary Material

4.6.1 Nextflow

The Nextflow package, version 0.22.0 is available as a free open-source package from https://github.com/nextflow-io/nextflow. It comes complete with documentation and examples. Nextflow is a domain-specific language (DSL) modelled around a functional-reactive programming (FRP) style and based on the *Dataflow* [187] paradigm. It uses a declarative processing model developed for parallel tasks execution in which concurrency and synchronization are managed automatically. Within Nextflow, biological pipelines are considered as a succession of tasks that must be independently declared and can then be combined into functional operators. When running a pipeline, the role of Nextflow is to deploy the tasks and insure their communication through dataflow variables. In order to declare the tasks and their relationships, Nextflow provides three important abstractions: *Processes*, *Channels* and *Operators*.

Processes are elements of code that define a unit of work. They are typically native commands, tools or scripts that implement specific stages of a pipeline (e.g. the RNA-Seq quantification command in Kallisto). They can be written in any scripting language supported by the target execution platform (BASH, Perl, Python, R, etc.). Each process is executed in a completely isolated manner and can only read and write data in its own uniquely assigned working directory, thereby preventing any interference across processes. Processes can either run natively, or within pre-specified Docker containers (see below). When submitted to the execution subsystem (workstation, cluster, cloud) these processes are enclosed in a wrapper script that manages the job submission and the collection of output files upon completion. Processes communicate with one another using **channels**. In a pipeline, with the exception of the main input/output, any given input process must be explicitly defined as the combination of one or more outputs generated by upstream processes. This hierarchy leads to an implicit definition of process dependencies as a directed acyclic graph. Channels are modelled as an asynchronous stream of data, behaving logically like UNIX pipes that would be able to stream any kind of arbitrarily structured data (i.e.

records, files, etc). The capacity to handle arbitrary units of information is important, as this makes it possible to manage the flows using **operators** able to perform basic operations such as merging, splitting and filtering. Operators may be described as a set of built-in processes, developed for the specific purpose of combining and manipulating channel's content. In the same way that they define channel combinations, operators also define process dependencies. For instance if an operator requires the merging of all the output from a given process, it implicitly defines the next stage as dependent on full completion of the previous stage. Under this very general model [188], a computational pipeline can be logically represented as a network of reactive processes in which each node waits for its inputs and it is automatically fired when all those inputs have been delivered. When doing so, the node produces new outputs that are either terminal, or ready to feed a downstream node.

Parallelization and queuing systems. Channel connections define process dependencies, and within a given flow, Nextflow processes are immune from race conditions thanks to their idempotency. Parallelization is implicitly defined by the way processes and operators are connected, thus allowing tasks to be simultaneously queued. Nextflow provides built-in support for the following batch schedulers: Open Grid Scheduler, Univa Grid Engine, Platform LSF, Linux SLURM, PBS Works, Torque and HTCondor. It also supports some emerging distributed computing platforms such as Apache Ignite and Kubernetes (beta). It includes built-in integration for cloud infrastructures such as Amazon AWS and DNAnexus. Nextflow is able to monitor individual tasks in order to skip redundant computations through a caching mechanism.

Error tolerance. Because of its computational model, Nextflow can efficiently handle errors and ease the computationally efficient debugging of large-scale computations. Upon unsuccessful termination of a process, Nextflow can re-submit the considered job a specified number of times, increasing the requested resources if needed, and eventually trigger a graceful exit. When this occurs, the maintenance of all intermediate temporary files makes it possible to track the issue and eventually resume computation where it stopped.

Versioning is a key aspect of reproducibility. Nextflow has been designed to integrate natively the Git tool for source code management and the most popular code repository services (GitHub,

BitBucket and GitLab). By default, Nextflow scans GitHub for any requested pipeline not locally available. The `-revision` option makes it possible to run any version of a given pipeline by specifying a Git branch, tag or commit ID defined in the project repository.

Built-in Docker support. Docker is a lightweight virtualization technology used to run applications in an isolated and self-contained environment. Containers only require a fraction of a second to start and have a negligible overhead cost on CPU and memory performance [179]. The main advantage of Docker is that it guarantees the full reproducibility of any piece of software by simply downloading a single pre-built, ready-to-run image containing all the software components and the required configuration. Docker images can either be stored as plain text configuration lists that insure full transparency, or as pre-built binaries that guarantee reproducibility even when the distribution package of a specific piece of software has become unavailable. In Nextflow, each process can be run within its own Docker container.

Multi Scale Containerization. Containers can be deployed at various levels. In the most naive approach the whole pipeline is executed within a container. This does not require any special support by Nextflow or any other tool, but it imposes a constraint on the pipeline scalability because the execution is limited by the computing resources available to a single container. It can also result in dead-end updates caused by incompatibility between tools. Nextflow implements a more granular approach, that we call multi-scale containerization, in which each pipeline stage is executed in a separate container instance. In this approach Nextflow orchestrates the container executions, which can be efficiently distributed in a cluster of computers with different stages of the pipeline executed simultaneously within different copies of the same container. A finer-grained approach is also supported by Nextflow. It involves using different container types for different stages. This allows the pipeline subcomponents to be packaged in separate container images thus isolating possible tools incompatibilities. Finally the container used by each pipeline step can be even defined dynamically. This allows, in a complex usage scenario, the dynamic reconfiguration of a task execution subcomponent with a different set of tools, depending on the task inputs context available at runtime. This approach can benefit a lot from the GitHub integration and provide a powerful framework for systematic benchmarking.

4.6.2 Companion analysis

Versions. Companion version 1.0.2 was forked from the original GitHub repository [189] into a new one available at this link https://github.com/cbcrg/companion. The *Leishmania infantum* dataset was included in the project repository along with a configuration file for each target execution platform. Each execution configuration was marked with a Git tag, namely `nbt-docker`, `nbt-macosx` and `nbt-awslinux`. This allowed us to run each experiment in a self-contained replicable manner, without requiring any undocumented steps, other than the installation of Companion dependencies in the target environment when the Docker engine was not used.

Platforms. The two platforms used for both native and Docker-based executions were an early-2014 MacBook Air running OSX Yosemite (10.10) and an Amazon cloud instance running Amazon Linux (AMI 2016.03). The Docker image is based on the Debian *stretch* Linux distribution.

Datasets. The *Leishmania infantum* genome (clone JPCM5, version 3.0a) was downloaded from Sanger. The default reference *Leishmania major* data included in the Companion repository was used as the reference dataset for the annotation. Other databases used included Pfam Version 30.0 and the core Gene Ontology (GO) file (version 'releases/2016-09-07') which are included in the Docker image. The dataset has been archived on Zenodo (see reproducibility section).

Commands. For the native Mac OSX execution i.e. without Docker, the Companion dependencies were installed in the target computer and the pipeline was executed with the following command:

```
$ nextflow run cbcrg/companion -revision nbt-macosx
```

The above command downloads and runs from the GitHub repository the project tree labeled with the `nbt-macosx` tag which contains the required data and platform specific configuration (see Reproducibility section).

For native Linux executions, as before, the pipeline dependencies were installed in the target computer, then the execution was launched with the following command:

```
$ nextflow run cbcrg/companion -revision nbt-awslinux
```

The above command downloads and runs the pipeline from GitHub using the `nbt-awslinux` source tree that contains the Linux specific configuration.

Finally we repeated the executions in the same environments by using the Docker engine. The pipeline was executed in both cases by using the following command:

```
$ nextflow run cbcrg/companion -revision nbt-docker
```

This command downloads and runs the pipeline. It does so by automatically pulling the Docker image specified in the configuration file from the public repository provided by the original authors (see Reproducibility section).

Stability. In order to rule out any fluctuation that may result from some random sampling procedure, each analysis was repeated at least two times on each platform. In each instance the behavior was found to be entirely deterministic on each individual platform.

Reproducibility. The pipeline source code along with the experiment results have been archived in the Zenodo storage service and they are accessible with the following DOI number: 10.5281/zenodo.154520. The Companion pipeline is also available directly from the GitHub repository along with detailed instruction how to replicate this experiment: https://github.com/cbcrg/companion. The Docker image contains all the Companion dependencies, thus making it unnecessary to install them in the target environment. In this distribution, the Docker image used for this computation is referenced in the Nextflow configuration file using its uniquely generated SHA256 identifier. This guarantees that image content will never change over time.

4.6.3 Kallisto and Sleuth analysis

Versions. Kallisto (v0.42.4) was downloaded from https://pachterlab.github.io/kallisto and Sleuth (v0.28.0) from https://pachterlab.github.io/sleuth. The Nextflow version of the pipeline is named Kallisto-NF (Supp. Figure 2).

Platforms. The two platforms used for both native and Docker-based executions were an early-2014 MacBook Air running OSX Yosemite (10.10) and an Amazon cloud instance running Amazon Linux (AMI 2016.03). The Docker image is based on the Debian *jessie* Linux distribution.

Datasets. The RNA-Seq dataset is taken from the Kallisto and Sleuth 'Getting Started' tutorials and based on the original Cuffdiff2 publication [190]. The six samples were downloaded from the NCBI short reads archive. These reads correspond to human lung fibroblasts transfected with either HOXA1 directed siRNA or non-targeting 'scramble' siRNA. All samples are paired-end 100 bp reads sequenced with an Illumina HiSeq 2000. The transcriptome Homo_sapiens.GRCh38.rel79.cdna.all.fa.gz was downloaded from the Kallisto website with the human gene annotation from Ensembl (version 79). All raw data has been uploaded to Zenodo (see reproducibility section).

Commands. The native BASH version of Kallisto and Sleuth pipeline was run using the following command line:

```
$ kallisto-std.sh data/reads/ data/transcriptome.fa data/experiment.txt
kallisto ./results
```

See Supplementary Figure 1 for the listing of the `kallisto-std.sh` script. The Nextflow counterpart was run using:

```
$ nextflow run kallisto.nf --reads 'data/reads/*.fastq' --transcriptome
data/transcriptome.fa --experiment data/experiment.txt -with-docker
```

An outline of the source code of `kallisto.nf` is shown Figure 2. Note that the Nextflow version, although a bit longer than the BASH implementation, can handle any number of read pairs, run the quantification step in parallel and is portable across different clusters and clouds.

Stability. In order to rule out any fluctuation that may result from some random sampling procedure, each analysis was repeated at least two times on each platform. In each instance the behavior was found to be entirely deterministic on each individual platform.

Reproducibility. The pipeline source code along with the experiment results have been archived in the Zenodo storage service and they are accessible with the following DOI number: 10.5281/zenodo.159195. The input dataset has been stored on Zenodo as well and is accessible with the following DOI number: 10.5281/zenodo.159158. The Docker image specified in the configuration file from the public repository is available at https://hub.docker.com/r/cbcrg/kallisto-nf. This image contains all the Kallisto and Sleuth dependencies, thus making it unnecessary to install them in the target environment. In this distribution, the Docker image used for this computation is referenced in the Nextflow configuration file using it's uniquely generated SHA256 identifier. This guarantees that image content will never change over time. The complete instructions along with links to the input dataset needed to replicate this experiment are available in the GitHub repository at the following link https://github.com/cbcrg/kallisto-nf-reproduce/tree/nbt-v1.0.

4.6.4 RAxML analysis

Versions. RAxML (v8.0.0) was downloaded from https://github.com/stamatak/standard-RAxML/archive/v8.0.0.zip. For each of the native executions and the Docker version, RAxML was compiled using the default make settings:

```
$ make -f Makefile.gcc
```

Platforms. The two platforms used for both native and Docker-based executions were an early-2014 MacBook Air running OSX Yosemite (10.10) and an Amazon cloud instance running Amazon Linux (AMI 2016.03). The Docker image is based on the Debian *wheezy* Linux distribution.

Datasets. The dataset used contains motor protein Prestin from 35 different species, as published by Liu et al. [191]. Motor protein Prestin is expressed in mammalian outer hair cells (OHCs) and is thought to confer high frequency sensitivity and selectivity in the mammalian auditory system.

Commands.

Tree computation. RAxML was natively run using the following command lines:

Native (Mac OSX and Linux)

```
$ nextflow run cbcrg/raxml-nf -r nbt-v1.0
```

The counterpart enabling the Docker execution was run using:

```
$ nextflow run cbcrg/raxml-nf -r nbt-v1.0 -with-docker
```

Tree comparison. The tree comparison was performed using ETE Toolkit v3.0.0b35 [192], that can be downloaded from the following link: http://etetoolkit.org/download/. The command used was the following:

```
$ ete3 compare -t input_tree_1 -r input_tree_2 --unrooted
```

Stability. In order to rule out any fluctuation that may result from some random sampling procedure, each analysis was repeated at least two times on each platform. In each instance the behavior was found to be entirely deterministic on each individual platform.

Reproducibility. The pipeline source code along with the input dataset and the experiment results have been archived in the Zenodo storage service and they are accessible with the following DOI number: 10.5281/zenodo.159181. The pipeline is also available directly from the GitHub repository along with detailed instructions on how to replicate this experiment: https://github.com/cbcrg/raxml-nf/tree/nbt-v1.0.

4.6.5 Supplementary Tables

Supplementary Table 1. Comparison between Nextflow and other state-of-art workflow management systems.

Workflow	Nextflow	Galaxy	Toil	Snakemake	Bpipe
Platform	Groovy/JVM	Python	Python	Python	Groovy/JVM
Native Task Support	Any	No	No	BASH only	BASH only
Common Workflow Language	No	Yes	Yes	No	No
Streaming processing	Yes	No	No	No	No
Dynamic branch evaluation	Yes	?	Yes	Yes	undoc.
Code Sharing Integration	Yes	No	No	No	No
Workflow modules	No	Yes	Yes	Yes	Yes
Workflow versioning	Yes	Yes	No	No	No
Automatic error failover	Yes	No	Yes	No	No
Graphical interface	No	Yes	No	No	No
DAG rendering	Yes	Yes	Yes	Yes	Yes
Container management	**Nextflow**	**Galaxy**	**Toil**	**Snakemake**	**Bpipe**
Docker support	Yes	Yes	Yes	No	No
Shifter support	Yes	No	No	No	No
Multi-scale containers	Yes	Yes	Yes	No	No
Built-in batch schedulers	**Nextflow**	**Galaxy**	**Toil**	**Snakemake**	**Bpipe**
Univa Grid Engine	Yes	Yes	Yes	Partial	Yes
PBS/Torque	Yes	Yes	No	Partial	Yes
LSF	Yes	Yes	No	Partial	Yes
SLURM	Yes	Yes	Yes	Partial	No
HTCondor	Yes	Yes	No	Partial	No

Built-in distributed cluster	Nextflow	Galaxy	Toil	Snakemake	Bpipe
Apache Ignite	Yes	No	No	No	No
Apache Spark	No	No	Yes	No	No
Kubernetes	Yes	No	No	No	No
Apache Mesos	No	No	Yes	No	No
Built-in cloud	Nextflow	Galaxy	Toil	Snakemake	Bpipe
AWS	Yes	Yes	Yes	No	No

This table provides an overview of the relative characteristic of the most widely used workflow management systems currently available. Comparison items include:

- *Workflow*
    - *Platform*: The technology and the programming language in which each framework is implemented.
    - *Native tasks support*: refers to the ability of the framework to support the execution of native commands and scripts without re-implementation of the original processes.
    - *Common Workflow Language*: Support for the CWL specification [9].
    - *Streaming processing*: Ability to process tasks inputs/outputs as a stream of data.
    - *Code sharing integration*: Support for code management and sharing platforms such as Github, Bitbucket and GitLab.
    - *Workflow modules*: Support for module, sub-workflows or workflow compositions.
    - *Workflow versioning*: Ability to track pipeline changes and to execute different version at any point in time.
    - *Automatic error failover*: Support for automatic error handling and resume execution mechanism.
    - *Graphical user interface:* Implementation of a graphical user interface to interact to with the pipeline.
    - *DAG rendering:* Ability to visualize the graph of the task dependencies and executions.

- ***Built-in support containers technology****:*
    - *Docker support*: Integrated support for Docker containers technology [10].
    - *Shifter support:* Integrated support for Shifter containers technology [11].
    - *Multi-scale containers:* Ability to manage the execution of multiple container instances in a distributed/HPC cluster or cloud.
- **Built-in support resource manager / batch scheduler:** Ability to spawn the executions of pipeline tasks through a cluster batch scheduler without the need of custom scripts or commands. It must be noted that although this support is not built-in for Snakemake, it merely requires the user to provide the cluster specific job control commands.
- **Built-in support for distributed cluster:** Ability to spawn the executions of pipeline tasks through a distributed cluster scheduler.
- **Built-in Cloud:** Support for cloud deployment.

Supplementary Table 2. Comparison of the Companion pipeline annotation of *Leishmania infantum* genome executed across different platforms natively (Mac OSX, Amazon Linux, Debian Linux) and with Docker (Mac OSX and Amazon Linux). All three native executions resulted in different annotations whilst Docker executions result in the same annotation across platforms.

Platform	**Mac OSX**	**Amazon Linux**	**Debian Linux**	**Mac OSX**	**Amazon Linux**
Execution	**Native**	**Native**	**Native**	**NF+Docker**	**NF+Docker**
number of chromosomes	36	36	36	36	36
overall length (bp)	32,032,223	32,032,223	32,032,223	32,032,223	32,032,223
number of genes	7,771	7,781	7,783	7,783	7,783
gene density	236.32	236.64	236.64	236.64	236.64
number of coding genes	7570	7,580	7,580	7,580	7,580
average coding length (bp)	1,762	1,764	1,764	1,764	1,764
number of genes with multiple CDS	111	113	113	113	113
number of genes with known function	4,142	4,147	4,147	4,147	4,147
number of t-RNAs	88	88	90	90	90

4.6.6 Supplementary figures

```
#!/bin/bash

# Raw Reads Directory = $1
# Transcriptome       = $2
# Experiment          = $3
# Kallisto Binary     = $4
# Output Directory    = $5

# Index the transcriptome
bin/$4 index -i $5/human_GRCh38.idx $2

# Run Kallisto Quantification for each set of fastq files
bin/$4 quant -i $5/human_GRCh38.idx -o $5/kallisto/SRR493366 -b 100 $1/SRR493366_1.fastq $1/SRR493366_2.fastq
bin/$4 quant -i $5/human_GRCh38.idx -o $5/kallisto/SRR493367 -b 100 $1/SRR493367_1.fastq $1/SRR493367_2.fastq
bin/$4 quant -i $5/human_GRCh38.idx -o $5/kallisto/SRR493368 -b 100 $1/SRR493368_1.fastq $1/SRR493368_2.fastq
bin/$4 quant -i $5/human_GRCh38.idx -o $5/kallisto/SRR493369 -b 100 $1/SRR493369_1.fastq $1/SRR493369_2.fastq
bin/$4 quant -i $5/human_GRCh38.idx -o $5/kallisto/SRR493370 -b 100 $1/SRR493370_1.fastq $1/SRR493370_2.fastq
bin/$4 quant -i $5/human_GRCh38.idx -o $5/kallisto/SRR493371 -b 100 $1/SRR493371_1.fastq $1/SRR493371_2.fastq

# Run Sleuth
bin/sleuth.R $5/kallisto $3
```

Supplementary Figure 1. (10.5281/zenodo.159164) Kallisto Native Pipeline. The Kallisto native pipeline is written in bash and calls Kallisto to perform indexing of the transcriptome, RNA-seq pseudo-mapping and quantification. It then calls Sleuth to perform differential expression analysis.

```
/* https://github.com/cbcrg/kallisto-nf */
/* Paolo Di Tommaso <paolo.ditommaso@crg.eu> */
/* Evan Floden <evan.floden@crg.eu> */

params.transcriptome = "$baseDir/tutorial/transcriptome/transcriptome.fa"
params.name          = "RNA-Seq Abundance Analysis"
params.reads         = "$baseDir/tutorial/reads/*.fastq"
params.fragment_len  = '180'
params.fragment_sd   = '20'
params.bootstrap     = '100'
params.experiment    = "$baseDir/tutorial/experiment/hiseq_info.txt"
params.output        = "results/"

transcriptome_file     = file(params.transcriptome)
exp_file               = file(params.experiment)

Channel
    .fromFilePairs( params.reads, size: -1 )
    .ifEmpty { error "Cannot find any reads matching: ${params.reads}" }
    .set { read_files }

process index {
    input:
    file transcriptome_file
    output:
    file "transcriptome.index" into transcriptome_index
    script:
    """
    kallisto index -i transcriptome.index ${transcriptome_file}
    """
}

process mapping {
    tag "reads: $name"
    input:
    file transcriptome_index from transcriptome_index.first()
    set val(name), file(reads) from read_files
    output:
    file "kallisto_${name}" into kallisto_out_dirs
    script:
    def single = reads instanceof Path
    if( !single ) {
        """
        mkdir kallisto_${name}
        kallisto quant -b ${params.bootstrap} -i transcriptome.index -t ${task.cpus} -o kallisto_${name} ${reads}
        """
    }
    else {
        """
        mkdir kallisto_${name}
        kallisto quant --single -l ${params.fragment_len} -s ${params.fragment_sd} -b ${params.bootstrap} \
                -i ${transcriptome_index} -t ${task.cpus} -o kallisto_${name} ${reads}
        """
    }
}

process sleuth {
    input:
    file 'kallisto/*' from kallisto_out_dirs.toSortedList()
    file exp_file
    output:
    file 'sleuth_object.so'
    file 'gene_table_results.txt'
    script:
    """
    sleuth.R kallisto ${exp_file}
    """

}
```

Supplementary Figure 2. (10.5281/zenodo.159166) Kallisto Nextflow pipeline. The native Kallisto pipeline is converted to Nextflow and composed of three processes. The first two processes call Kallisto to index the transcriptome and then pseudo-map for RNA-seq quantification, and the third one uses Sleuth to perform differential expression analysis.

Supplementary Figure 3. (10.5281/zenodo.159168) Interleaved output of two RAxML Phylogenetic Trees of the same sequences estimated on Mac OSX (blue) and Amazon Linux (red). Differences in the branch lengths of resulting trees are shown in color. No such differences were observed when running a Nextflow dockerized version of the same command.

5. Discussion

One of the main and most recent challenges of modern biology is to keep-up with the growing amount of biological data produced by experiments using next generation sequencing technologies. New sources of high throughput data are also about to emerge, especially with respect to automated phenotypic and behavioral analysis. Making sense of this data will require a significant scale up of available methods. Large-scale comparative bioinformatics analyses are an integral part of this procedure and their better integration with high performance technologies will soon be an essential part of the biological endeavor.

When doing comparative bioinformatics, multiple sequence alignments (MSAs) are by far the most widely used modeling methods [1], with the publication describing ClustalW [2] pointing at #10 among the most cited scientific papers of all time. Indeed, a large number of in-silico analyses depend on multiple sequence alignment methods. These include domain analysis, phylogenetic reconstruction, motif finding and a whole range of other applications, extensively described in [3 - 4].

Multiple Sequence Alignment is an important modeling tool whose development has required addressing a very complex combination of computational and biological problems. The computation of an accurate MSA has long been known to be an NP-complete problem, a situation that explains why over 100 alternative methods have been developed these last three decades [4] for the alignment of evolutionarily related sequences, while taking into account evolutionary events such as mutations, insertions, deletions, and rearrangement under certain conditions. In this thesis we extensively reviewed progress made over the last decade that include the development of consistency based methods, the development of sequence/structure alignment methods, the development of structure-based RNA aligners and the development of index-based filtering methods. We concluded that the main challenges for multiple sequence aligners when catching up with growing datasets will not be limited to improving or maintaining reasonable levels of accuracy. It will also involve the proper quantification of readouts reliability. In fact, one may argue that this quantification is possibly more important than the overall accuracy, at least at the levels achieved by existing methods.

We demonstrated in this thesis that alignment uncertainty, which results in multiple alignment methods generating different MSAs whenever the input-order of sequences changes, is an

inherent property of the progressive framework. We established that about two-thirds of this uncertainty stems from unordered nature of children nodes within the guide trees used to estimate MSAs. We showed that all currently available large-scale multiple alignment methods are numerically unstable and produce significantly different output when changing sequence input-order. On datasets larger than 100 sequences, this instability affects on average 21% of the residues when considering the most stable aligners, and grows worse with higher number of sequences to align and higher evolutionary distance.

Phylogenetic reconstructions are essential in genomics comparative data analyses and depend on accurate multiple sequence alignment (MSA) models. Thus, the resulting Maximum Likelihood trees estimated from these multiple sequence alignments are equally unstable with over 38% of the branches being sensitive to the sequence input-order. To quantify this uncertainty we developed shootstrap, a novel approach that estimates the combined effect of alignment uncertainty and evolutionary sampling on phylogenetic tree branch supports. This reliability index is more informative than the standard bootstrap procedure when estimating phylogenetic confidence intervals. So far shootstrap has only been validated on protein sequences but we expect the instability that it helps quantifying to be higher in RNA and DNA sequences whose lower complexity alphabet results in a larger number of alternative optimal alignments. We intend to explore shootstrap behavior on nucleic acids as an immediate follow up of the work presented here. Yet, even when applied onto protein alignments, shootstrap high sensitivity in detecting poorly supported branches gives it the potential to improve many downstream analyses, including attempts to predict the effect of genome variations on molecular functions [152].

Shootstrap is a method geared towards solving some of the problems alignment uncertainty causes in phylogenetic analyses. Shootstrap, however, merely estimates reliability but does not help improving the phylogenetic models. To achieve this, one would need to improve the original MSAs accuracy. New MSA methods should not only become more stable but also more accurate.

MEGA-Coffee is a novel and highly accurate method for producing stable and biological meaningful alignments of any number of sequences. We argue here that the MEGA-Coffee, accuracy and its scaling up capacities are essential to cope with the current trend of data analyses, ultimately leading to higher-quality biological insights. MEGA-Coffee outperforms existing alignment methods on large and/or difficult-to-align data sets in both terms of speed and accuracy. At the same time MEGA-Coffee is less affected by alignment uncertainty compared to existing

methods. This makes MEGA-Coffee a more stable aligner that guarantees delivery of the same high quality alignments, regardless of sequence input-order, ensuring thus better reproducibility of data and biological interpretations.

To perform large-scale comparative bioinformatics analyses, complex pipelines that consist of multiple third-party software, which have many dependencies on external scripts, libraries, environmental variables etc, have to be constructed. These pipelines, due to the amount of data they have to deal with, require high performance computing (HPC) resources, thus they often have to be shipped from a laptop, to a powerful desktop computer, to a cluster or even a cloud or a supercomputing center. Due to these reasons and a combination of factors hard to control for, reproducing routine bioinformatics analysis is challenging.

Nextflow, was specifically designed to address this problem. It is a flow management framework that uses container technology to insure efficient deployment and reproducibility of computational analysis pipelines. Third party pipelines can be ported into nextflow with minimum re-coding. We used RNA-Seq quantification and phylogeny reconstruction examples to show how two seemingly irreproducible analyzes can be made stable across platforms when ported into Nextflow. Its low level makes it compatible with higher level solutions, such as Galaxy [180] which provides a convenient and powerful interface to prototype pipelines, but only offers limited support for their parallelization [224]. Even though the current beta-version of Nextflow has been online for a short amount of time, several institutions have already ported their pipelines into this framework, including Nextflow Workbench at Cornell University [184], the Companion suite for small pathogen annotation at the Sanger [173], the Needlestack pipeline by the International Agency for Research on Cancer, and the functional annotation of the Phytozome data pipeline at Joint Genome Institute. This rapidly growing community of users reflects well on the pressing need addressed by Nextflow and on its capacity to blend very efficiently into ongoing projects. At a time when technology keeps evolving at a breathtaking pace, Nextflow solves one of the few technical issues bound to be long lasting, both in the academic and in the hospital world: the quest for computational reproducible large-scale analysis.

References

1. Van Noorden R, Maher B, Nuzzo R. The top 100 papers. Nature 2014; 514:550–3
2. Thompson J, Higgins D, Gibson T. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 1994; 22:4673–4690
3. Thompson JD, Linard B, Lecompte O, et al. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. PLoS One 2011; 6:e18093
4. Kemena C, Notredame C. Upcoming challenges for multiple sequence alignment methods in the high-throughput era. Bioinformatics 2009; 25:2455–2465
5. Edgar RC, Batzoglou S. Multiple sequence alignment. Curr Opin Struct Biol 2006; 16:368–373
6. Notredame C, Higgins DG. SAGA: sequence alignment by genetic algorithm. Nucleic Acids Res 1996; 24:1515–24.
7. Hogeweg P, Hesper B. The alignment of sets of sequences and the construction of phylogenetic trees. An integrated method. J. Mol. Evol. 1984; 20:175–186
8. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 1970; 48:443–453
9. Notredame C, Higgins DG, Heringa J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. J Mol Biol 2000; 302:205–17.
10. Do CB, Mahabhashyam MS, Brudno M, et al. ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Res 2005; 15:330–340
11. Wallace IM, O'Sullivan O, Higgins DG. Evaluation of iterative alignment algorithms for multiple alignment. Bioinformatics 2005; 21:1408–1414
12. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 2004; 32:1792–7. Print 2004.
13. Katoh K, Misawa K, Kuma K, et al. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res 2002; 30:3059–66.
14. Sievers F, Wilm A, Dineen D, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 2011; 7:539
15. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 1987; 4:406–425
16. Murtagh F. Complexities of hierarchic clustering algorithms: State of the art. 1984; 1:
17. Wheeler TJ, Kececioglu JD. Multiple alignment by aligning alignments. Bioinformatics 2007; 23:i559-68
18. Morgenstern B, Frech K, Dress A, et al. DIALIGN: finding local similarities by multiple sequence alignment. Bioinformatics 1998; 14:290–294

19. Kececioglu JD. The maximum weight trace problem in multiple sequence alignment. Lect. Notes Comput. Sci. 1983; 684:106–119
20. Kececioglu JD, Lenhof H-P, Mehlhorn K, et al. A polyhedral approach to sequence alignment problems. Discret. Appl. Math. 2000; 104:143–186
21. Paten B, Herrero J, Beal K, et al. Sequence progressive alignment, a framework for practical large-scale probabilistic consistency alignment. Bioinformatics 2009; 25:295–301
22. Durbin R, Eddy S, Krogh A, et al. Biological Sequence Analysis. 1998;
23. Liu Y, Schmidt B, Maskell DL. MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. Bioinformatics 2010; 26:1958–1964
24. Di Tommaso P, Orobitg M, Guirado F, et al. Cloud-Coffee: implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud. Bioinformatics 2010; 26:1903–1904
25. Rausch T, Emde AK, Weese D, et al. Segment-based multiple sequence alignment. Bioinformatics 2008; 24:i187-92
26. Breen MS, Kemena C, Vlasov PK, et al. Epistasis as the primary factor in molecular evolution. Nature 2012; 490:535–538
27. Mirarab S, Nguyen N, Guo S, et al. PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences. J. Comput. Biol. 2015; 22:377–86
28. Liu K, Raghavan S, Nelesen S, et al. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science (80-.). 2009; 324:1561–1564
29. Edgar RC. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 2004; 5:113
30. Blackshields G, Sievers F, Shi W, et al. Sequence embedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol. Biol. 2010; 5:21
31. Löytynoja A, Goldman N. An algorithm for progressive multiple alignment of sequences with insertions. Proc. Natl. Acad. Sci. U. S. A. 2005; 102:10557–62
32. Morrison DA. Why would phylogeneticists ignore computerized sequence alignment? Syst. Biol. 2009; 58:150–8
33. Blackburne BP, Whelan S. Class of multiple sequence alignment algorithm affects genomic analysis. Mol. Biol. Evol. 2013; 30:642–53
34. Markova-Raina P, Petrov D. High sensitivity to aligner and high rate of false positives in the estimates of positive selection in the 12 Drosophila genomes. Genome Res. 2011; 21:863–74
35. Chang JM, Tommaso P, Notredame C. TCS: A new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. Mol. Biol. Evol. 2014;
36. Rost B. Twilight zone of protein sequence alignments. Protein Eng 1999; 12:85–94
37. Chang JM, Di Tommaso P, Taly JF, et al. Accurate multiple sequence alignment of transmembrane proteins with PSI-Coffee. BMC Bioinformatics 2012; 13 Suppl 4:S1

38. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. J Mol Biol 1999; 292:195–202.
39. Pei J, Kim BH, Grishin N V. PROMALS3D: a tool for multiple protein sequence and structure alignments. Nucleic Acids Res 2008; 36:2295–2300
40. O'Sullivan O, Suhre K, Abergel C, et al. 3DCoffee: combining protein sequences and structures within multiple sequence alignments. J Mol Biol 2004; 340:385–95.
41. Armougom F, Moretti S, Poirot O, et al. Expresso: automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee. Nucleic Acids Res 2006; 34:W604-8
42. Capriotti E, Marti-Renom MA. Quantifying the relationship between sequence and three-dimensional structure conservation in RNA. BMC Bioinformatics 2010; 11:322
43. Sankoff D. Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM J. Appl. Math. 1985; 45:810–825
44. Dowell RD, Eddy SR. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. BMC Bioinformatics 2006; 7:400
45. Mathews DH, Turner DH. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. J. Mol. Biol. 2002; 317:191–203
46. Mathews DH. Predicting a set of minimal free energy RNA secondary structures common to two sequences. Bioinformatics 2005; 21:2246–53
47. Holmes I. Accelerated probabilistic inference of RNA structure evolution. BMC Bioinformatics 2005; 6:73
48. Gorodkin J, Heyer LJ, Stormo GD. Finding the most significant common sequence and structure motifs in a set of RNA sequences. Nucleic Acids Res. 1997; 25:3724–32
49. Havgaard JH, Lyngso RB, Stormo GD, et al. Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. Bioinformatics 2005; 21:1815–1824
50. Tabei Y, Kiryu H, Kin T, et al. A fast structural multiple alignment method for long RNA sequences. BMC Bioinformatics 2008; 9:33
51. Tabei Y, Tsuda K, Kin T, et al. SCARNA: fast and accurate structural alignment of RNA sequences by matching fixed-length stem fragments. Bioinformatics 2006; 22:1723–9
52. McCaskill JS. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers 1990; 29:1105–1119
53. Kiryu H, Tabei Y, Kin T, et al. Murlet: a practical multiple alignment tool for structural RNA sequences. Bioinformatics 2007; 23:1588–1598
54. Siebert S, Backofen R. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. Bioinformatics 2005; 21:3352–3359
55. Hofacker IL, Bernhart SHF, Stadler PF. Alignment of RNA base pairing probability matrices. Bioinformatics 2004; 20:2222–7
56. Will S, Reiche K, Hofacker IL, et al. Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. PLoS Comput Biol 2007; 3:e65

57. Torarinsson E, Havgaard JH, Gorodkin J. Multiple structural alignment and clustering of RNA sequences. Bioinformatics 2007; 23:926–932
58. Will S, Joshi T, Hofacker IL, et al. LocARNA-P: accurate boundary prediction and improved detection of structural RNAs. RNA 2012; 18:900–14
59. Sorescu DA, Möhl M, Mann M, et al. CARNA--alignment of RNA structure ensembles. Nucleic Acids Res. 2012; 40:W49-53
60. Do CB, Foo C-S, Batzoglou S. A max-margin model for efficient simultaneous alignment and folding of RNA sequences. Bioinformatics 2008; 24:i68-76
61. Harmanci AO, Sharma G, Mathews DH. Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. BMC Bioinformatics 2007; 8:130
62. Will S, Otto C, Miladi M, et al. SPARSE: quadratic time simultaneous alignment and folding of RNAs without sequence-based heuristics. Bioinformatics 2015; 31:2489–96
63. Dalli D, Wilm A, Mainz I, et al. STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. Bioinformatics 2006; 22:1593–9
64. Bauer M, Klau GW, Reinert K. Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. BMC Bioinformatics 2007; 8:271
65. Xu X, Ji Y, Stormo GD. RNA Sampler: a new sampling based algorithm for common RNA secondary structure prediction and structural alignment. Bioinformatics 2007; 23:1883–91
66. Reeder J, Giegerich R. Consensus shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction. Bioinformatics 2005; 21:3516–3523
67. Wilm A, Higgins DG, Notredame C. R-Coffee: a method for multiple alignment of non-coding RNA. Nucleic Acids Res 2008; 36:e52
68. Bernhart SH, Hofacker IL, Stadler PF. Local RNA base pairing probabilities in large sequences. Bioinformatics 2006; 22:614–615
69. Katoh K, Toh H. Improved accuracy of multiple ncRNA alignment by incorporating structural information into a MAFFT-based framework. BMC Bioinformatics 2008; 9:212
70. Dror O, Nussinov R, Wolfson H. ARTS: alignment of RNA tertiary structures. Bioinformatics 2005; 21 Suppl 2:ii47-53
71. Capriotti E, Marti-Renom MA. RNA structure alignment by a unit-vector approach. Bioinformatics 2008; 24:i112-8
72. Ferrè F, Ponty Y, Lorenz WA, et al. DIAL: a web server for the pairwise alignment of two RNA three-dimensional structures using nucleotide, dihedral angle and base-pairing similarities. Nucleic Acids Res. 2007; 35:W659-68
73. Rahrig RR, Leontis NB, Zirbel CL. R3D Align: global pairwise alignment of RNA 3D structures using local superpositions. Bioinformatics 2010; 26:2689–97
74. Chang Y-F, Huang Y-L, Lu CL. SARSA: a web tool for structural alignment of RNA using a structural alphabet. Nucleic Acids Res. 2008; 36:W19-24
75. Bauer RA, Rother K, Moor P, et al. Fast Structural Alignment of Biomolecules Using a Hash

Table, N-Grams and String Descriptors. Algorithms 2009; 2:692–709
76. Kemena C, Bussotti G, Capriotti E, et al. Using tertiary structure for the computation of highly accurate multiple RNA alignments with the SARA-Coffee package. Bioinformatics 2013; 29:1112–1119
77. Seibel PN, Müller T, Dandekar T, et al. 4SALE--a tool for synchronous RNA sequence and secondary structure alignment and editing. BMC Bioinformatics 2006; 7:498
78. Lück R, Gräf S, Steger G. ConStruct: a tool for thermodynamic controlled prediction of conserved secondary structure. Nucleic Acids Res. 1999; 27:4208–17
79. Jeon Y-S, Chung H, Park S, et al. jPHYDIT: a JAVA-based integrated environment for molecular phylogeny of ribosomal RNA sequences. Bioinformatics 2005; 21:3171–3
80. Griffiths-Jones S. RALEE--RNA ALignment editor in Emacs. Bioinformatics 2005; 21:257–9
81. Andersen ES, Lind-Thomsen A, Knudsen B, et al. Semiautomated improvement of RNA alignments. Rna 2007; 13:1850–1859
82. Derrien T, Johnson R, Bussotti G, et al. The GENCODE v7 catalog of human long noncoding RNAs: Analysis of their gene structure, evolution, and expression. Genome Res 2012; 22:1775–1789
83. Bussotti G, Raineri E, Erb I, et al. BlastR--fast and accurate database searches for non-coding RNAs. Nucleic Acids Res 2011; 39:6886–6895
84. Lindgreen S, Gardner PP, Krogh A. MASTR: multiple alignment and structure prediction of non-coding RNAs using simulated annealing. Bioinformatics 2007; 23:3304–3311
85. Orom UA, Derrien T, Beringer M, et al. Long noncoding RNAs with enhancer-like function in human cells. Cell 2010; 143:46–58
86. Tilgner H, Nikolaou C, Althammer S, et al. Nucleosome positioning as a determinant of exon recognition. Nat. Struct. Mol. Biol. 2009; 16:996–1001
87. Sankoff D, Blanchette M. Multiple genome rearrangement and breakpoint phylogeny. J Comput Biol 1998; 5:555–70.
88. Dewey CN. Aligning multiple whole genomes with Mercator and MAVID. Methods Mol. Biol. 2007; 395:221–36
89. Angiuoli S V, Salzberg SL. Mugsy: fast multiple alignment of closely related whole genomes. Bioinformatics 2011; 27:334–42
90. Brudno M, Do CB, Cooper GM, et al. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. Genome Res 2003; 13:721–731
91. Blanchette M, Kent WJ, Riemer C, et al. Aligning multiple genomic sequences with the threaded blockset aligner. Genome Res 2004; 14:708–715
92. Paten B, Herrero J, Beal K, et al. Enredo and Pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. Genome Res. 2008; 18:1814–28
93. Raphael B, Zhi D, Tang H, et al. A novel method for multiple alignment of sequences with repeated and shuffled elements. Genome Res 2004; 14:2336–2346

94. Paten B, Earl D, Nguyen N, et al. Cactus: Algorithms for genome multiple sequence alignment. Genome Res. 2011; 21:1512–28
95. Kehr B, Trappe K, Holtgrewe M, et al. Genome alignment with graph data structures: a comparison. BMC Bioinformatics 2014; 15:99
96. Earl D, Nguyen N, Hickey G, et al. Alignathon: a competitive assessment of whole-genome alignment methods. Genome Res 2014; 24:2077–2089
97. Kim J, Ma J. PSAR: measuring multiple sequence alignment reliability by probabilistic sampling. Nucleic Acids Res 2011; 39:6359–6368
98. Su J, Teichmann SA, Down TA. Assessing computational methods of cis-regulatory module prediction. PLoS Comput. Biol. 2010; 6:e1001020
99. Aerts S. Computational strategies for the genome-wide identification of cis-regulatory elements and transcriptional targets. Curr. Top. Dev. Biol. 2012; 98:121–45
100. Berezikov E, Guryev V, Plasterk RHA, et al. CONREAL: conserved regulatory elements anchored alignment algorithm for identification of transcription factor binding sites by phylogenetic footprinting. Genome Res. 2004; 14:170–8
101. Sinha S, He X. MORPH: probabilistic alignment combined with hidden Markov models of cis-regulatory modules. PLoS Comput Biol 2007; 3:e216
102. Satija R, Pachter L, Hein J. Combining statistical alignment and phylogenetic footprinting to detect regulatory elements. Bioinformatics 2008; 24:1236–42
103. Satija R, Novák A, Miklós I, et al. BigFoot: Bayesian alignment and phylogenetic footprinting with MCMC. BMC Evol. Biol. 2009; 9:217
104. Ettwiller L, Paten B, Souren M, et al. The discovery, positioning and verification of a set of transcription-associated motifs in vertebrates. Genome Biol. 2005; 6:R104
105. Majoros WH, Ohler U. Modeling the evolution of regulatory elements by simultaneous detection and alignment with phylogenetic pair HMMs. PLoS Comput. Biol. 2010; 6:e1001037
106. Erb I, Gonzalez-Vallinas JR, Bussotti G, et al. Use of ChIP-Seq data for the design of a multiple promoter-alignment method. Nucleic Acids Res 2012; 40:e52
107. Notredame C, Holm L, Higgins DG. COFFEE: an objective function for multiple sequence alignments. Bioinformatics 1998; 14:407–22.
108. Edgar RC. Quality measures for protein alignment benchmarks. Nucleic Acids Res
109. Iantorno S, Gori K, Goldman N, et al. Who watches the watchmen? An appraisal of benchmarks for multiple sequence alignment. Methods Mol. Biol. 2014; 1079:59–73
110. Bahr A, Thompson JD, Thierry JC, et al. BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. Nucleic Acids Res 2001; 29:323–326
111. Van Walle I, Lasters I, Wyns L. SABmark--a benchmark for sequence alignment that covers the entire known fold space. Bioinformatics 2005; 21:1267–1268
112. Sievers F, Dineen D, Wilm A, et al. Making automated multiple alignments of very large

numbers of protein sequences. Bioinformatics 2013; 29:989–95
113. Lackner P, Koppensteiner WA, Sippl MJ, et al. ProSup: a refined tool for protein structure alignment. Protein Eng 2000; 13:745–52.
114. Armougom F, Moretti S, Keduas V, et al. The iRMSD: a local measure of sequence alignment accuracy using structural information. Bioinformatics 2006; 22:e35-9
115. Gardner PP, Wilm A, Washietl S. A benchmark of multiple sequence alignment programs upon structural RNAs. Nucleic Acids Res. 2005; 33:2433–9
116. Yilmaz P, Parfrey LW, Yarza P, et al. The SILVA and 'All-species Living Tree Project (LTP)' taxonomic frameworks. Nucleic Acids Res. 2014; 42:D643-8
117. Ogden TH, Rosenberg MS. Multiple sequence alignment accuracy and phylogenetic inference. Syst. Biol. 2006; 55:314–28
118. Stoye J, Evers D, Meyer F. Rose: generating sequence families. Bioinformatics 1998; 14:157–63
119. Strope CL, Abel K, Scott SD, et al. Biological sequence simulation for testing complex evolutionary hypotheses: indel-Seq-Gen version 2.0. Mol. Biol. Evol. 2009; 26:2581–93
120. Cartwright RA. DNA assembly with gaps (Dawg): simulating sequence evolution. Bioinformatics 2005; 21 Suppl 3:iii31-8
121. Fletcher W, Yang Z. INDELible: a flexible simulator of biological sequence evolution. Mol. Biol. Evol. 2009; 26:1879–88
122. Wallace IM, Blackshields G, Higgins DG. Multiple sequence alignments. Curr Opin Struct Biol 2005; 15:261–266
123. Hall BG. Comparison of the accuracies of several phylogenetic methods using protein and DNA sequences. Mol. Biol. Evol. 2005; 22:792–802
124. Rosenberg MS. Multiple sequence alignment accuracy and evolutionary distance estimation. BMC Bioinformatics 2005; 6:278
125. Wang L-S, Leebens-Mack J, Kerr Wall P, et al. The impact of multiple protein sequence alignment on phylogenetic estimation. IEEE/ACM Trans. Comput. Biol. Bioinform. 8:1108–19
126. Dessimoz C, Gil M. Phylogenetic assessment of alignments reveals neglected tree signal in gaps. Genome Biol. 2010; 11:R37
127. O'Sullivan O, Zehnder M, Higgins D, et al. APDB: a novel measure for benchmarking sequence alignment methods without reference alignments. Bioinformatics 2003; 19 Suppl 1:i215-21
128. Lin K, Kleinjung J, Taylor WR, et al. Testing homology with Contact Accepted mutatiOn (CAO): a contact-based Markov model of protein evolution. Comput. Biol. Chem. 2003; 27:93–102
129. Kemena C, Taly JF, Kleinjung J, et al. STRIKE: evaluation of protein MSAs using a single 3D structure. Bioinformatics 2011; 27:3385–3391
130. Hickson RE, Simon C, Perrey SW. The performance of several multiple-sequence alignment programs in relation to secondary-structure features for an rRNA sequence. Mol Biol Evol 2000;

17:530–9.
131. Talavera G, Castresana J. Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments. Syst Biol 2007; 56:564–577
132. Capella-Gutierrez S, Silla-Martinez JM, Gabaldon T. trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. Bioinformatics 2009; 25:1972–1973
133. Pei J, Grishin N V. AL2CO: calculation of positional conservation in a protein sequence alignment. Bioinformatics 2001; 17:700–12.
134. Thompson JD, Thierry JC, Poch O. RASCAL: rapid scanning and correction of multiple sequence alignments. Bioinformatics 2003; 19:1155–1161
135. Mevissen HT, Vingron M. Quantifying the local reliability of a sequence alignment. Protein Eng. 1996; 9:127–32
136. Landan G, Graur D. Heads or tails: a simple reliability check for multiple sequence alignments. Mol Biol Evol 2007; 24:1380–1383
137. Penn O, Privman E, Landan G, et al. An alignment confidence score capturing robustness to guide tree uncertainty. Mol Biol Evol 2010; 27:1759–1767
138. Sela I, Ashkenazy H, Katoh K, et al. GUIDANCE2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters. Nucleic Acids Res. 2015; 43:W7–W14
139. Chang J-M, Di Tommaso P, Notredame C. TCS: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. Mol. Biol. Evol. 2014; 31:1625–37
140. Jarvis ED, Mirarab S, Aberer AJ, et al. Whole-genome analyses resolve early branches in the tree of life of modern birds. Science (80-.). 2014; 346:1320–1331
141. Wong KM, Suchard MA, Huelsenbeck JP. Alignment uncertainty and genomic analysis. Science (80-.). 2008; 319:473–476
142. Pittis AA, Gabaldón T. Late acquisition of mitochondria by a host with chimaeric prokaryotic ancestry. Nature 2016; 531:101–4
143. Krypotou E, Evangelidis T, Bobonis J, et al. Origin, diversification and substrate specificity in the family of NCS1/FUR transporters. Mol. Microbiol. 2015; 96:927–50
144. Vandewege MW, Mangum SF, Gabaldón T, et al. Contrasting Patterns of Evolutionary Diversification in the Olfactory Repertoires of Reptile and Bird Genomes. Genome Biol. Evol. 2016; 8:470–80
145. Sankoff D, Cedergren RJ. Simultaneous comparison of three or more sequences related by a tree. Time warps, string Ed. Macromol. theory Pract. Seq. Comp. 1983; 253–263
146. Wang L, Jiang T. On the complexity of multiple sequence alignment. J. Comput. Biol. 1994; 1:337–348
147. Price MN, Dehal PS, Arkin AP. FastTree: computing large minimum evolution trees with profiles instead of a distance matrix. Mol. Biol. Evol. 2009; 26:1641–50

148. Anisimova M, Gascuel O. Approximate likelihood-ratio test for branches: A fast, accurate, and powerful alternative. Syst. Biol. 2006; 55:539–52
149. Felsenstein J. Confidence limits on phylogenies: an approach using the bootstrap. Evolution (N. Y). 1985; 39:783–791
150. Boyce K, Sievers F, Higgins DG. Simple chained guide trees give high-quality protein multiple sequence alignments. Proc. Natl. Acad. Sci. 2014; 111:10556–10561
151. Tan G, Gil M, Löytynoja AP, et al. Simple chained guide trees give poorer multiple sequence alignments than inferred trees in simulation and phylogenetic benchmarks. Proc. Natl. Acad. Sci. U. S. A. 2015; 112:E99-100
152. Katsonis P, Lichtarge O. A formal perturbation equation between genotype and phenotype determines the Evolutionary Action of protein-coding variations on fitness. Genome Res. 2014; 24:2050–8
153. Chatzou M, Magis C, Chang J-M, et al. Multiple sequence alignment modeling: methods and applications. Brief. Bioinform. 2015;
154. Nguyen N-PD, Mirarab S, Kumar K, et al. Ultra-large alignments using phylogeny-aware profiles. Genome Biol. 2015; 16:124
155. Holm L, Sander C. Removing near-neighbour redundancy from large protein sequence collections. Bioinformatics 1998; 14:423–9
156. Farrar M. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. Bioinformatics 2007; 23:156–61
157. Zhao M, Lee W-P, Garrison EP, et al. SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications. PLoS One 2013; 8:e82138
158. Soding J. Protein homology detection by HMM-HMM comparison. Bioinformatics 2005; 21:951–960
159. Thompson JD, Koehl P, Ripp R, et al. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. Proteins 2005; 61:127–136
160. Magis C, Taly JF, Bussotti G, et al. T-Coffee: Tree-based consistency objective function for alignment evaluation. Methods Mol Biol 2014; 1079:117–129
161. Roshan U, Livesay DR. Probalign: multiple sequence alignment using partition function posterior probabilities. Bioinformatics 2006; 22:2715–2721
162. Allison DB, Brown AW, George BJ, et al. Reproducibility: A tragedy of errors. Nature 2016; 530:27–9
163. . Reviewing computational methods. Nat. Methods 2015; 12:1099–1099
164. . Rebooting review. Nat. Biotechnol. 2015; 33:319–319
165. Goodman SN, Fanelli D, Ioannidis JPA. What does research reproducibility mean? Sci. Transl. Med. 2016; 8:
166. Baker M. Muddled meanings hamper efforts to fix reproducibility crisis. Nature 2016;
167. Masca NG, Hensor EM, Cornelius VR, et al. RIPOSTE: a framework for improving the design

and analysis of laboratory-based research. Elife 2015; 4:
168. Piccolo SR, Frampton MB. Tools and techniques for computational reproducibility. Gigascience 2016; 5:30
169. LeVeque RJ, Mitchell IM, Stodden V. Reproducible research for scientific computing: Tools and strategies for changing the culture. Comput. Sci. Eng. 2012; 14:13–17
170. Garijo D, Kinnings S, Xie L, et al. Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome. PLoS One 2013; 8:e80278
171. Loman N, Watson M. So you want to be a computational biologist? Nat. Biotechnol. 2013; 31:996–8
172. Bray NL, Pimentel H, Melsted P, et al. Near-optimal probabilistic RNA-seq quantification. Nat. Biotechnol. 2016; 34:525–7
173. Steinbiss S, Silva-Franco F, Brunk B, et al. Companion : a web server for annotation and analysis of parasite genomes. Nucleic Acids Res. 2016; 44:W29–W34
174. Belmann P, Dröge J, Bremges A, et al. Bioboxes: standardised containers for interchangeable bioinformatics software. Gigascience 2015; 4:47
175. Moreews F, Sallou O, Ménager H, et al. BioShaDock: a community driven bioinformatics shared Docker-based tools registry. F1000Research 2015; 4:1443
176. Hosny A, Vera-Licona P, Laubenbacher R, et al. AlgoRun: a Docker-based packaging system for platform-agnostic implemented algorithms. Bioinformatics 2016; 32:2396–8
177. Perkel J. Democratic databases: science on GitHub. Nature 2016; 538:127–128
178. Köster J, Rahmann S. Snakemake--a scalable bioinformatics workflow engine. Bioinformatics 2012; 28:2520–2
179. Di Tommaso P, Palumbo E, Chatzou M, et al. The impact of Docker containers on the performance of genomic pipelines. PeerJ 2015; 3:e1273
180. Goecks J, Nekrutenko A, Taylor J, et al. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 2010; 11:R86
181. Blankenberg D, Von Kuster G, Bouvier E, et al. Dissemination of scientific software with Galaxy ToolShed. Genome Biol. 2014; 15:403
182. Vivian J, Rao A, Nothaft FA, et al. Rapid and efficient analysis of 20,000 RNA-seq samples with Toil. bioRxiv 2016;
183. Liu K, Linder CR, Warnow T. RAxML and FastTree: Comparing two methods for large-scale maximum likelihood phylogeny estimation. PLoS One 2011; 6:
184. Kurs JP, Simi M, Campagne F. NextflowWorkbench: Reproducible and Reusable Workflows for Beginners and Experts. bioRxiv 2016;
185. Di Tommaso P. A curated list of Nextflow pipelines.
186. Byron SA, Van Keuren-Jensen KR, Engelthaler DM, et al. Translating RNA sequencing into clinical diagnostics: opportunities and challenges. Nat. Rev. Genet. 2016; 17:257–71

187. Johnston WM, Hanna JRP, Millar RJ. Advances in Dataflow Programming Languages.
188. Lee EA, Parks TM. DATAFLOW PROCESS NETWORKS. Proc. IEEE 1995; 773–801
189. Companion. https://github.com/sanger-pathogens/companion.
190. Trapnell C, Hendrickson DG, Sauvageau M, et al. Differential analysis of gene regulation at transcript resolution with RNA-seq. Nat. Biotechnol. 2012; 31:46–53
191. Liu Y, Cotton JA, Shen B, et al. Convergent sequence evolution between echolocating bats and dolphins. Curr. Biol. 2010; 20:R53-4
192. Huerta-Cepas J, Serra F, Bork P. ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data. Mol. Biol. Evol. 2016; 33:1635–8