New algorithmic contributions for large scale multiple sequence alignments of protein sequences

Edgar Garriga Nogales

Tesi Doctoral UPF / 2021
Universitat Pompeu Fabra - Barcelona

DIRECTOR DE LA TESI
Dr. Cédric Notredame

BIOINFORMATICS AND GENOMICS PROGRAMME
CENTRE FOR GENOMIC REGULATION (CRG)

DEPARTMENT OF EXPERIMENTAL AND HEALTH SCIENCES
UNIVERSITAT POMPEU FABRA

Acknowledgments

The last four years have been challenging, a path of learning and feeding curiosity.

This adventure has been incredible, thanks to the priceless help of Cedric Notredame. None of this would have been possible without him.

Four years of maturity, of exploring new worlds and enjoying this experience with those who share time or knowledge with me.

The environment created at the CRG and the facilities to focus only on your work are also appreciated, a unique location with incredible human and material resources. Being able to work in a place where it offers to satisfy your curiosity is the best thing that has happened to me. I would like to add some words to Evan and Paolo, to be always there. To Leila and Athanasios to suffer with me this adventure. And last but not least, Romina, if a Ph.D. is hard, I don't want to imagine how this would be without her. From traveling to printing or organizing seminars.

Thanks to Cedric and friends of this trip. For allowing me to combine this adventure with the world outside the laboratory. Sometimes it is needed to go outside this bubble and see the real world.

I don't want to forget all the brave people who risk their lives trying to cross a border. They teached me things I would not be able to learn by reading books or manuscripts.

Finally, I also want to thank my parents for supporting me on this path, which has not been easy or quiet. They allowed me to study and go abroad whenever I asked for it. Without them, their sacrifice and support would never be possible. Gracias

"Si los derechos que yo tengo no lo tienen los demás,
entonces no son derechos, son privilegios"

Abstract

A multiple sequence alignment (MSA) describes the connection between biological sequences by using columns to represent a shared ancestry based on an assumed set of evolutionary events.

Keeping up with the growth of biological data is one of modern biology's most relevant and recent challenges.

Unfortunately, MSA approaches have a well-known weakness when dealing with extremely large datasets that could endanger any downstream analysis. Therefore, the most common approach to the MSA analysis is the progressive solution.

Another of the challenges in the MSAs due to the rapid increase of data is the scalability of the alignments. That's why a new methodology will be proposed to perform the MSAs with the ability of higher scalability and the possibility of deciding how the guide-tree is used, and the combination of guide-tree and alignment works better for each study.

This study, taken together, can be considered as providing both conceptual and technical innovations that significantly improve existing MSA methods and a new paradigm.

Resum

Els alineaments multiples de sequencies descriuen la conexio entre sequencies biologiques fent us de les columnes per representar una ascendencia en comu basat en la asumpcio de uns fets evolutius.

Mantenir-se al dia amb la creixent de les dades biològiques és un dels reptes més rellevants i recents de la biologia moderna.

Malauradament, els enfocaments MSA tenen una debilitat ben coneguda quan es tracta de treballar amb conjunts de dades extremadament grans que poden posar en perill qualsevol anàlisi posterior. L'enfocament més comú en l'anàlisi MSA és la metodologia progressiva.

Un altre dels reptes en els MSA degut al ràpid augment de dades és l'escalabilitat dels alineaments. És per això que aquí es proposarà una nova metodologia per realitzar els MSA amb la capacitat d'una major escalabilitat i la possibilitat de decidir com s'utilitza el "guide-tree" i la combinació "guide-tree" i alineament que funcioni millor per a cada cas.

Aquest estudi, en conjunt, es pot considerar que proporciona innovacions tant conceptuals com tècniques que milloren significativament els mètodes MSA existents i un nou paradigma.

Preface

The document is organized so that the introduction chapter provides background information for readers who may not be familiar with multiple sequence alignment methods and a small introduction to the following topics of this thesis. Following that, there is a manuscript in each of the three key chapters. The reader will discover further background information for the individual subject covered in each chapter here. Except for the introductory and discussion chapters, each chapter's references, figures, and tables are self-contained to avoid confusion. The purpose of the discussion chapter is to put the outcomes of each chapter into context.

Contents

Chapter 1: Introduction

Comparisons provide us with some of our most fundamental biological insights. Moreover, it is not without reason. Comparative biology takes advantage of the single most significant fact in biology: that all life is interconnected. Through the fabric of our developments, relatedness has remained the core principle, weaved with ever-changing technologies. The concept of shared ancestry is at the heart of most of our biology, and when Darwin created the iconic tree of life, he originally conceived homology through anatomical comparisons, such as those of bones, beaks, and barnacles. Comparative anatomy was also valuable for quantifying the degree of similarity across species.

Sequences play a particular role in bioinformatics and therefore deserve considerable attention. For example, our measures of similarity are quantified at single-molecule resolution using a sequence. Nevertheless, more critically, the sequence representation enabled biology to stand on the shoulders of computer scientists and mathematicians by providing a toolkit of quantification tools. Strings of characters lend themselves to computation in a variety of ways, from simple edit distances to complicated machine learning techniques.

Sequence comparison is one of the most extensively studied fields of computer science, where it is generally referred to as string matching (Apostolico & Galil, 1997; Baeza-Yates & Navarro, 1996, 1998; Boytsov, 2011; Gusfield, 1997; G. Myers, 1999). The necessity of matching sequences expressed as strings is so pressing that as early as 1970, biologists felt the urge to develop their own algorithm and came out with an independent formulation of the Dijkstra dynamic programming algorithm (Dijkstra, 1959). This algorithm, commonly referred as Needleman and Wunsch (Needleman & Wunsch, 1970), remains the cornerstone of evolutionary-based sequence alignments in biology. It is worth noting, however, noting that this string matching approach was recently complemented by a new generation of string matching techniques built on the Burrow Wheeler transform (Burrows & Wheeler, 1994). However, these highly efficient algorithms remain limited to non-evolutionary ultra-fast string matching, such as the one required for mapping reads onto a genome or carrying out assemblies.

Over the years, the Needleman and Wunsch (NW) algorithm has undergone many significant evolutions. When first introduced, it was meant to align sequences that one

expected to be homologous across their entire length, yet, with new sequences coming in, it became rapidly clear that more complex matching schemas were required taking into account the frequent need for partial matches, that to say aligning two sequences whose homology does not span the full length. This adaptation of NW is known as the Smith and Waterman algorithm (Smith & Waterman, 1981) and uses a modified version of NW to identify the highest-scoring common segment between two homologous sequences. The solution initially devised by Needleman and Wunsch is generalizable using Levenshtein's approach, which aims to reduce edit distance (Sellers, 1974).

The main idea behind Needleman-Wunsch is that it effectively eliminates comparisons that cannot contribute to the best possible score alignment. The algorithm was then improved significantly in terms of performance and memory utilization. A significant breakthrough was the Gotoh algorithm that provides a way to estimate the matching cost in linear space and in quadratic time (Gotoh, 1982). Gotoh also provided the first quadratic solution for estimating alignments with affine gap penalty schemes, that is to say, gaps whose cost could be modeled as $cost = Gap\ Opening\ Penalty + Gap\ Extension\ Penalty * Length.$ Under Gotoh, the actual resolution of the alignments remained quadratic in space, thus limiting the algorithm to relatively short sequences. A few years later, Myers and Miller (E. W. Myers & Miller, 1988), building on the connection that had been made between Needleman and Wunsch and the Dijkstra algorithm, proposed a divide-and-conquer linear space algorithm for the alignment computation. This algorithm adapts Hirschberg (Hirschberg, 1975) to recursively compute middle points until the complete alignment has been achieved. Each middle point is obtained by applying the Gotoh algorithm in a forward and backward manner.

Multiple sequence alignment was a natural follow-up of pairwise sequence alignments. The problem of aligning a set of evolutionarily related sequences was initially theorized by Sankoff (Sankoff, 1975), who proposed a complete albeit impractical algorithm for the simultaneous estimation of an MSA and its underlying phylogenetic tree. Unfortunately, this problem is NP-Hard under any realistic formulation, and the Sankoff formulation remains limited to very few short sequences. Even when the evolutionary relationship between the sequences is provided (i.e., phylogenetic tree), the problem of assembling an MSA that would minimize the evolutionary cost remains NP-Hard under its most common

formulations (J. Kececioglu, 1993; J. D. Kececioglu et al., 2000; Wang & Jiang, 1994). This probably explains why such a large number of heuristics have been applied to the MSA problem. The NP-Hard nature of the problem also explains the rapid deprecation of heuristics, whose sensitivity to sequence number and length is a recurring problem. The genesis of these algorithms have been extensively reviewed in (Chatzou et al., 2016; Edgar & Batzoglou, 2006; Kemena & Notredame, 2009; Thompson et al., 2011), and going through them in detail would be beyond the scope of this introduction. We will simply provide a couple of milestones. Today, the most commonly used algorithms such as the Clustal series, T-Coffee (Notredame et al., 2000), or Mafft (Katoh et al., 2002) rely on the progressive algorithm. This algorithm that involves aligning the sequences two by two following the order imposed by a guide tree was initially described by Paula Hoggeweg and later re-implemented in a variety of software, the most popular being by far the ClustalW (Thompson et al., 1994) algorithm that ranking in the position 10 of the most widely cited scientific papers ever (Van Noorden et al., 2014).

The explicit goal of an MSA approach is to align a set of biological sequences (RNA, proteins, and DNA) in a way that reflects their evolutionary, structural, or functional relationship. This is accomplished by adding gaps of varying lengths inside the sequences, allowing homologous places to be aligned with one another, much to how an abacus aligns beads of the same hue. These gaps in the genome are assumed to result from insertions and deletions (indels) that occurred during the evolution of a common ancestor in an evolutionary environment.

A scoring function (objective function) capable of quantifying the relative benefits of every alternative alignment concerning the modeled relationship is required to construct an MSA. After that, the MSA can be calculated using an optimal scoring model. Thus, the objective function is a crucial parameter since it specifies the modeling accuracy and prediction capacity of an MSA. The most common goal functions used in evolutionary reconstructions are to maximize weighted similarities (as supplied by a PAM (Dayhoff & Schwartz, 1978) or BLOSUM (Henikoff & Henikoff, 1992) substitution matrix) while calculating indels costs using an affine gap penalty.

The substitution cost can be changed using tree-based weighting systems that reflect each sequence's independent information contribution, and the column score is derived by taking into account the overall all-against-all (sums-of-pairs) substitution cost. The sum-of-pairs functions are widely recognized for their inability to reflect biological

relationships precisely. However, they have been demonstrated to give a good trade-off between structural accuracy and computability, that is, the ability to estimate a reasonable MSA quickly.

The optimization of sums-of-pairs evaluation systems is NP-complete in their most popular formulations. As a result, one must rely on heuristics. One of the first solutions proposed is Hogeweg and Hesper's progressive alignment algorithm.

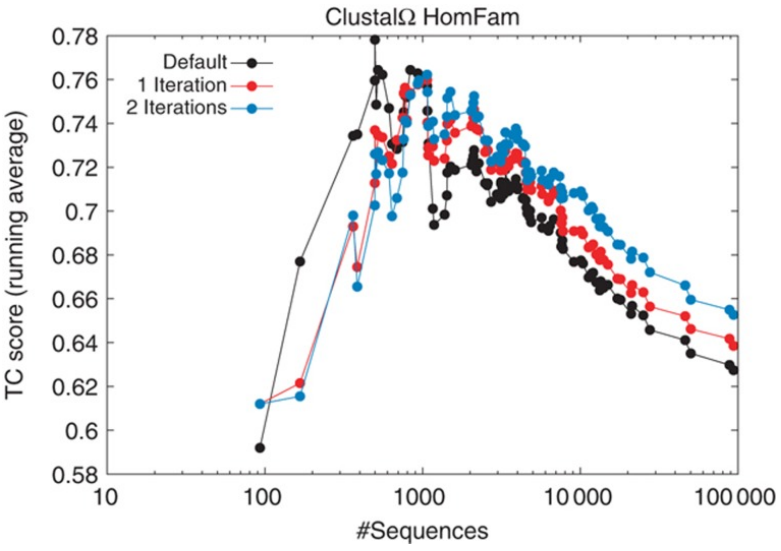
The input sequences are incorporated into the final model one by one, following an inclusion order determined by a pre-computed guide tree. Then a pairwise alignment is performed at each node between two sequences, a sequence and a profile, or two profiles. More or less sophisticated modifications of the Needlman and Wunsch global dynamic programming alignment algorithm are used to estimate the pairwise alignments at each node. The backbone of most accessible approaches, such as T-Coffee, ClustalW, and ProbCons (Do et al., 2005), combines a tree-based progressive strategy and a global pairwise alignment algorithm.

The guide tree estimation technique is the key algorithmic component of the progressive alignment, aside from the objective function. This tree, which determines which sequences will be included in which order, can be generated using a variety of methods, the most common of which being Neighbor-Joining (NJ) (Saitou & Nei, 1987) and Unweighted Pair Group Method with Arithmetic Mean (UPGMA) (Michener & Sokal, 1957).

In theory, an MSA would have to be evaluated based on a true underlying phylogenetic tree. If such a tree were available, it would then become possible to estimate a model minimizing the number of mutational events accounting for the differences among the sequences being considered. However, this tree is not usually available, and the most commonly used scoring schemes involve an all against all comparison known as Sums of Pairs. However, initially speculated by Hogeweg that aligning the sequences two by two following an order as close as possible to the correct phylogenetic tree could lead to reasonable solutions. This heuristic is purely greedy and does not provide any guarantee. It is well known to be sensitive to local minima effects. Furthermore, the correctness of the final MSA depends on the correctness of the initial guide tree. Moreover, even if this tree is correct, our limited capacity to match distantly related sequences can hamper the alignment process. Indeed the pairwise algorithm accuracy drops rapidly when matching protein sequences less than 20% identical (or DNA sequences less than 80% identical) (Rost,

1999), as a consequence, if the guide tree directs the alignment of two very distantly related sequences, the gap insertion patterns are likely to be incorrect and will possibly lead to severe effects when incorporating the remaining sequences. This problem could only be alleviated if one could simultaneously consider all the sequences, a formulation known to be NP-Hard. There exist two alternatives to address this issue. The most natural one is using an iterative approach where the guide tree is re-estimated on the first MSA. This strategy has been implemented in a large variety of algorithms, and albeit it clearly leads to improvements, these tend to be limited. The second alternative is the use of consistency. The consistency-based algorithm, initially described by Notredame, Higgins and Heringa (Notredame et al., 2000), aims to collect dataset-wide information to generate a position-specific scoring scheme considering the entire dataset, even when doing a pairwise alignment of any of its members. This algorithm currently forms the basis of the most accurate small-scale aligners (T-Coffee, Probcons, MSAProb (Liu et al., 2010)).

Until about ten years ago, when the largest datasets would hardly feature more than 500 sequences, these heuristics were able to provide entirely satisfying MSA models. However, the surge in sequencing capacity has brought these algorithms to their limits. The tilting point was probably the analysis carried out in ClustalO (Sievers et al., 2011), in which it was shown for the first time that the alignment accuracy decreases when aligning 1,000 sequences or more. This result, later confirmed by one of the studies presented here (Garriga et al., 2019), came as a shock as it suggested a behavior totally opposite to the expectation. It was also challenging the notion that acquiring more data would lead to richer, more informative models.



Sievers et al., 2011, Mol Syst Bio

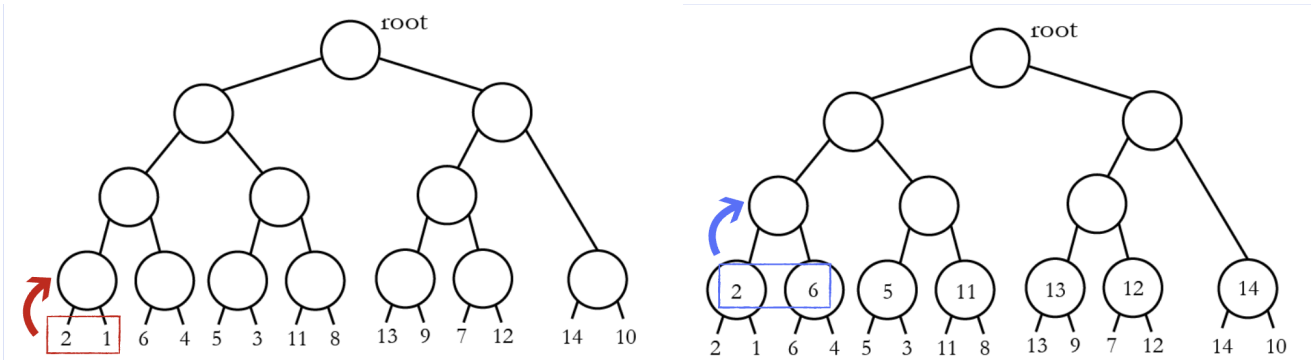
There are numerous reasons for the scaling up to fail so drastically. The first is most likely the accuracy of the guide-tree. The most widely used distance-based tree-building algorithms (NJ and UPGMA), which determine the order in which sequences are aligned, have computational complexity ranging from $O(N^2)$ to $O(N^3)$, depending on the implementation. As a result, these methods become impractical beyond a few thousand sequences, and generating a guide-tree with 100,000 sequences would necessitate the computation of nearly 5 billion distances.

Reducing the number of comparisons conducted is the most obvious technique to decrease tree building time and memory needs. This was initially effectively applied with the PartTree algorithm (Katoh & Toh, 2007), which selects a subset of sequences and clusters them recursively. The longest sequence, the sequence with the lowest resemblance to the longest, and n - 2 random sequences are selected starting at the top and then at each recursion level, where n is the group size defined by the user. The remaining non-seed sequences are then paired with one of the seed sequences to form a new group, and the seed sequences are utilized to construct a UPGMA tree. The same technique is repeated for each group until all sequences have reached the tree's leaf. Then, the extended trees can be used to make the final tree. This leads to a significant increase in speed and a reduction in time complexity to quasilinear O (N log N). PartTree is able to align 60,000 sequences in a matter of minutes, according to the authors, using a conventional desktop.
Another guide-tree method that avoids full distance matrix calculation is the mBed algorithm (Blackshields et al., 2010)(Blackshields et al., 2010). mBed starts by selecting a set of seed sequences, similar to PartTree, but this time based on a constant stride selection from the length sorted dataset. These seed sequences can be refined or not, but the distance between each sequence and the reference sites is determined in either case. These distances are then transformed into a vector for each sequence, containing the reference sites' coordinates. The vectors are approximations for sequence distance, allowing us to develop an embedded distance matrix that may be used to build UPGMA guide-trees. In addition, for big datasets, k-means clustering can be used to cluster the vectors directly without the need for an embedded distance matrix (over 100,000 sequences). The software package for Clustal Omega contains mBed (Sievers et al., 2013).

Nevertheless, these large-scale guide tree-based methods all proved equally unable to scale up when evaluated in the Clustal Omega paper, and it can be speculated that the guide-tree was the culprit.

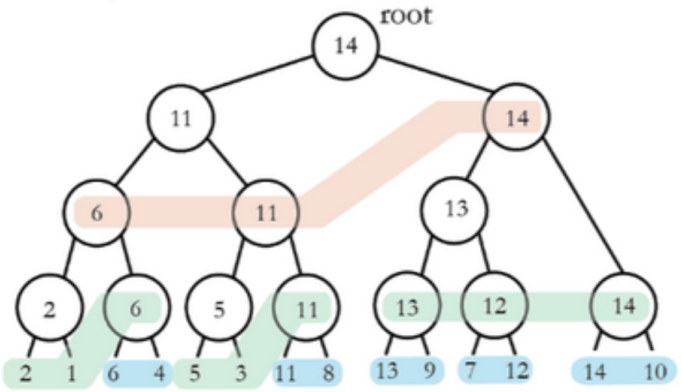
This led to a new generation of large-scale aligners in which, rather than being hierarchized and classified, the MSA is built from a small core of sequences gradually expanded. MAFFT Sparsecore (Yamada et al., 2016) was one of the first methods to use this strategy. The sequences are first sorted by length, and then a random selection of 500 sequences from the longest 50% of the sequences is chosen. These are the sequences that make up the core. The remaining sequences are added to the core using a progressive alignment method once the MSA is produced using the correct G-INS-i method. We later proposed a generalized version of this approach that we named the regressive alignment. This solution will be described in ***chapter 2***. It is based on a radically new way of using the guide-tree to resolve the final MSA

Its global strategy can be defined in 4 points. The first step consists of dressing the tree and filling the internal nodes with a "representative" sequence (by default, the longest among children leaves). This step is a bottom-up process.



The next step starts from the root and involves collecting subgroups of size N by collecting node representatives and expanding nodes one generation at a time. In other words, all nodes of a given generation are expanded into their children nodes before expanding any child. Expanding means replacing a node representative with the two representatives associated with its children. For example, in the figure below, N was set to 3, and one can see that the first set featured sequences 11 and 14. Because this set was smaller than N=3, node 11 was expanded into the (6, 11) set thus yielding the set (6, 11, and 14). If that set had featured less than N sequences, for instance, N=4, then node 14 would have been

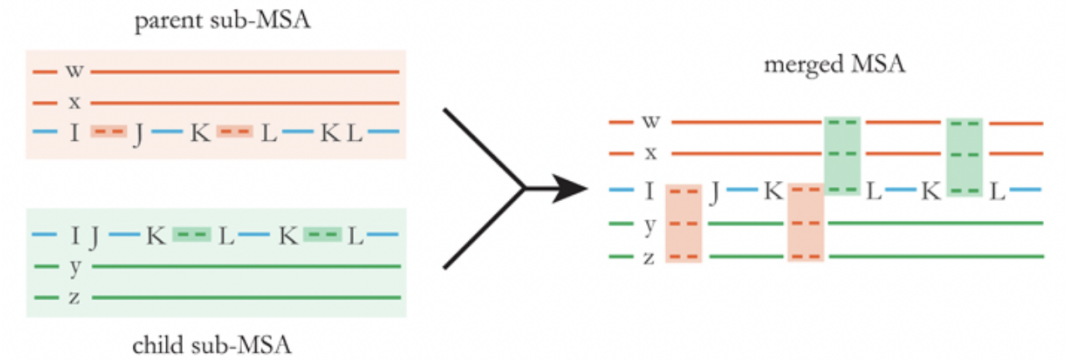
expanded into (13, 14). Once this first set has been gathered, each collected node is treated as a root so that an extra subset can be individually gathered. While this recursive process advances toward the root, the datasets become increasingly homogenous. The process stops when all the leaves have been collected.



Garriga et al., 2019

Once all the subgroups have been collected, the alignment step can take place. The independence of the subgroups makes the whole process an embarrassingly parallel operation. Furthermore, the homogenous size of the largest MSA makes it relatively easy to have optimally balanced loads. The process is entirely agnostic concerning the MSA method one should apply to the subgroups, and as we show in the paper, one is free to choose any method that would appear relevant. On top of this, the control one has on N makes it possible to adjust this value with the MSA method one wants to use (i.e., small values for slow accurate methods and higher values for coarser and faster aligners.

The fourth and final step involves the subMSAs. It does not require any alignment, thanks to the common sequences threading through the subgroups. For instance, one can see on the above figure that sequence 11 will be part of the root group (6,11,14), then that same sequence is part of the child group of this node (5,3,11) and is also part of the terminal child (11,8). Sequence 11 is therefore incorporated in these three independent MSAs, and it is suitable for the three MSAs to be threaded onto one another using the indexes of sequence 11 residues to match corresponding columns. This process does not require any kind of scanning or exploration as would be required by a standard alignment procedure since the equivalences are already set with no possible ambiguity. As such, it can be implemented in linear time with the length of the considered sequence. Furthermore, the memory footprint can be further decreased by storing the stretches of gaps as indexes rather than strings.



Garriga et al., 2019

The regressive algorithm is a versatile framework that lends itself to a wide variety of variations. First of all, it is not restricted to using guide trees, and any suitable hierarchical clustering approach could provide a support data structure. The most obvious would be hierarchical k-means, in which the centroids would play the role of nodes labels, but in theory, one could explore any similar clustering procedure. Secondly, the method lends itself to combining data at all possible levels, and with structural data now routinely available thanks to AF2 (Jumper et al., 2021), one could consider using structural aligners such as 3D-Coffee (O'Sullivan et al., 2004) to use structural information at various levels, and typically when aligning the root subset that is expected to be the most challenging owing to its high level of variability. Finally, the procedure would need to come along with a reliability index, similar to the TCS index developed in the realm of T-Coffee (Chang et al., 2014). At the scale targeted by the regressive algorithm, such indexes are essential for systematically removing non-trustworthy sequences. However, the high complexity of the TCS estimator precludes its systematic use in the regressive framework, and if that index is to be used, it will need to be adapted so as to allow the efficient flowing of lower scale TCS information onto a large number of sequences.

Nevertheless, before exploring these important algorithmic follow up, it was critical to ensure the usability and deployability of the regressive algorithm. Therefore, we initially implemented the method within the T-Coffee framework and made it part of the T-Coffee distribution. T-Coffee is both an aligner and a meta-aligner. Given a dataset, its alignment mode involves generating all the possible pairwise alignments using an internal Myers and Millers implementation. This collection is known as a library, and it is stored in a data structure that declares all the pairs of residues that have been found aligned in the collection of pairwise alignments. The consistency transformation is then used to turn this data structure into a position-specific scoring scheme. In broad terms, given two residues x

and y from sequences A and B, labeled by their index in their respective sequences, the consistency transformation involves collecting from the library all the supporting pairs x-z and z-y, with z being any residue from a third sequence that is neither A or B. These supporting pèairs are collected along with their non-supporting pairs, where a set of non-supporting pairs would be x-k and l-y, with k and l being different residues belonging to the same sequence. The supporting and non-supporting pairs can then be weighted, and the final consistency score for the alignment of x and y is obtained by combining all this information. This definition of T-Coffee corresponds to the latest implementation and differs slightly from the original publication in which the non-supporting pairs were not directly taken into account.

The main advantage of the library is its versatility. The lack of constraints regarding its completeness and redundancy means that any procedure able to populate it will be suitable for the computation of T-Coffee alignments. The most obvious adaptation involved generating the pairs with third-party MSA packages (Wallace et al., 2006) or even replacing the sequences with adequate protein structure templates (O'Sullivan et al., 2004) or RNA templates (Wilm et al., 2008), in which the pairwise alignments are carried out using suitable third party structural aligners. These developments that took place over nearly two decades gradually turned T-Coffee into a meta MSA package, and in the latest release, the program claims to be able to interact with a total of 73 third-party packages. Building the regressive algorithm in this framework allowed us to effectively use these existing connectors to effectively evaluate the regressive algorithm to combine various tree estimation methods with existing aligners.

Unfortunately, the original T-Coffee implementation had been carried out with single users in mind and with little provision for high-level parallelization. I, therefore, had to put significant effort into the design of portable and reproducible pipelines. For this purpose, I have used the Nextflow framework. Nextflow is built on containerization technology (like Docker (Merkel, 2014) or Singularity (Kurtzer et al., 2017)) or the package manager Conda. One of the examples can be found in ***chapter 3*** where I show how it was possible to build a portable environment for the regressive method and all its companion packages. By allowing the seamless installation of a large number of packages, most of which have been developed as research software rather than production software, I believe that our

approach has contributed to the democratization of computational bioinformatics, that is to say, making available a large number of alternative methods with less technical barriers.

These practices (containerization and reproducibility workflow managers) allow running the analysis independently of where one is based. The availability of cloud-based resources also makes it very straightforward to carry out complex development, even when the local economic status does not allow the maintenance of a complex IT infrastructure. In such a context, the software can be used to buffer some of the hardware limitations. These practices can save much money, and often enough, they are the only way to allow key analysis to be carried out locally. This manuscript shows how it is possible to install and use the regressive method to perform very large multiple sequence alignments independently of selected hardware. In the publication, we put much effort into ensuring that all the results could be reproduced using clear Nextflow command lines that would go and teach the software on GitHub and deploy it on data stored in Zenodo. For instance, the docker container with the environment configured and ready to run can be downloaded with the following command:

```
docker pull cbcrg/tcoffee_protocols
```

Then, once the image is in our system, you can run the Regressive algorithm with the TCoffee command:

```
t_coffee -reg -seq INPUT.fasta -reg_nseq 1000 -reg_tree mbed -reg_method clustalo_msa -outfile OUT.aln -outtree OUT.mbed
```

Furthermore, thanks to the workflow manager Nextflow, it is possible to run an analysis of Regressive using the containers provided (Docker, Conda, or Singularity) and run the pipeline in the local machine or move the computation to an HPC with the flag **-executor=slurm/sge** or the engine of your HPC system. If you want to launch the analysis directly to AWS cloud will be something like:

```
nextflow run cbcrg/dpa-analysis -executor=aws --seqs 'seqs/*.fasta'
```

A significant issue when doing large-scale computation is to keep track of data origin. For instance, if a diagnostic and treatment are proposed based on an exome analysis based on a

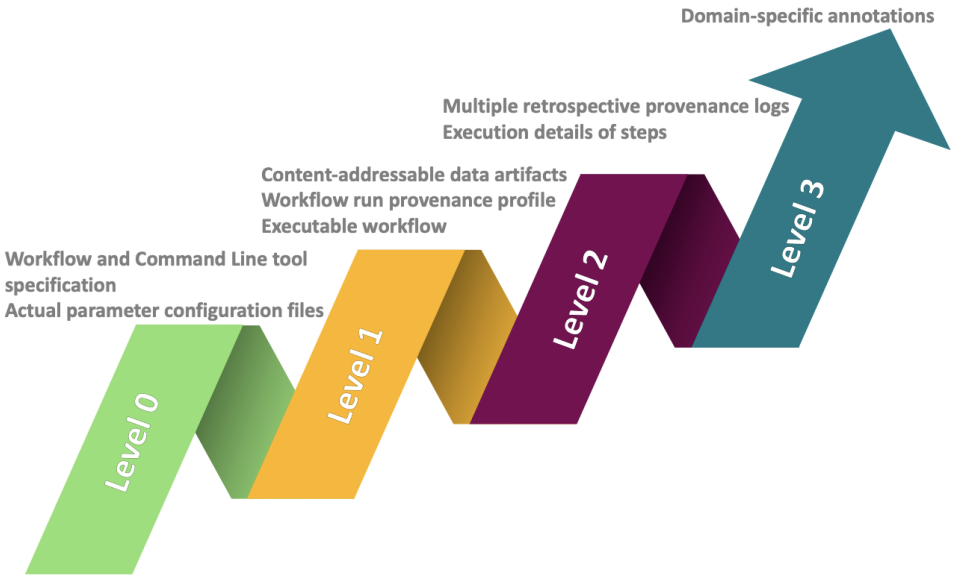
regressive MSA, one may ask where the MSA was carried out, which software version, and where the data was coming from. The traceability of these operations is a central engineering question, but it goes well beyond this. For example, if the treatment being offered appears to be sub-optimal, one should be entitled to know how the decision was reached to propose this treatment. This involves the decision-making process is based on open-source software. However, it also requires that software to be written in such a way that it be readable by trustworthy organizations, and even further, it also implies that any result having an impact on the life of a citizen should come along with some form of traceability, including an accurate description of the underlying resource. This issue is well known in computer science and relates to the concept of *provenance*, the notion that any computational result can be traced to its source.

Chapter 4 of this thesis reports our attempt to integrate Nextflow and the ResearchObject (RO) specification (Belhajjame et al., 2015). Our main goal was to implement in Nextflow the ability to produce a RO with a specific flag and add the extra information needed. This project has been supported by the Google Summer of Code (GSOC). The work is merely preliminary and involves the maintenance of informative tags passed by Nexftlow across successive computation steps. In its current form, it is not of direct use because it would require a suitable visualization tool for the practical exploitation of this meta-information. The idea of a need for provenance in the scientific analysis is not new (Davidson et al., 2007). This metadata adds more value and robustness to the pipeline.

This project triggered an interesting reflection in the Nextflow community and opened up the door for further development, mainly focused on provenance and its power with some discussion on the community (Ewels et al., 2020) to discuss how to deal with this implementation.

Adding an extra level allows the analysis to be more robust. The definition of provenance is "the history of ownership of a valued object or work of art or literature" and this is the main idea, recording the metadata and the valuable information of the analysis. This provenance aims to include who ran the pipeline, when it started and when it finished. All the software versions were used to perform the analysis. This feature will record the relationship between files (who created the file and what was the input data). This should be easier to track with the new version of Nextflow DSL2. The modularity is in the same idea of keeping the I/O data of each process and this can be used to debug the analysis and

detect anomalous behavior in the pipeline. This provenance will try to capture all the available information, which can be used to double-check parameters or who was responsible for the analysis.



Garriga et al., 2018

Chapter 2

Large multiple sequence alignments with a root-to-leaf regressive method

---

Edgar Garriga, Paolo Di Tommaso, Cedrik Magis, Ionas Erb, Leila Mansouri, Athanasios Baltzis, Hafid Laayouni, Fyodor Kondrashov, Evan Floden, Cedric Notredame

Nat Biotechnol 37, 1466–1470 (2019). https://doi.org/10.1038/s41587-019-0333-6

LETTERS
https://doi.org/10.1038/s41587-019-0333-6
nature biotechnology

Large multiple sequence alignments with a root-to-leaf regressive method

Edgar Garriga[1], Paolo Di Tommaso[1], Cedrik Magis[1], Ionas Erb[1], Leila Mansouri[1], Athanasios Baltzis[1], Hafid Laayouni[2,3], Fyodor Kondrashov[4], Evan Floden[1*] and Cedric Notredame[1,5*]

Multiple sequence alignments (MSAs) are used for structural[1,2] and evolutionary predictions[1,2], but the complexity of aligning large datasets requires the use of approximate solutions[3], including the progressive algorithm[4]. Progressive MSA methods start by aligning the most similar sequences and subsequently incorporate the remaining sequences, from leaf to root, based on a guide tree. Their accuracy declines substantially as the number of sequences is scaled up[5]. We introduce a regressive algorithm that enables MSA of up to 1.4 million sequences on a standard workstation and substantially improves accuracy on datasets larger than 10,000 sequences. Our regressive algorithm works the other way around from the progressive algorithm and begins by aligning the most dissimilar sequences. It uses an efficient divide-and-conquer strategy to run third-party alignment methods in linear time, regardless of their original complexity. Our approach will enable analyses of extremely large genomic datasets such as the recently announced Earth BioGenome Project, which comprises 1.5 million eukaryotic genomes[6].

Until the first benchmarking of large-scale MSAs, analyses made on smaller datasets suggested that scale-up would result in increased accuracy[7]. However, it has now been established that alignments with more than a thousand sequences are less accurate than smaller alignments[5]. It has been speculated[8] that this fall in accuracy is due to the inability of progressive methods to deal with the large number of gaps accumulated during intermediate alignment steps[9]. Recent attempts to address this problem have included SATé[10] and its follow-up PASTA[11,12], a progressive algorithm in which the guide tree is split into subsets that are independently aligned and later merged. This divide-and-conquer strategy allows computationally intensive methods to be deployed on large datasets but does not alleviate the challenge of merging very large intermediate MSAs. More recent alternatives include the MSA algorithms UPP[13] and MAFFT-Sparsecore[14] (Sparsecore). Both of these methods rely on selecting a subset of 'seed' sequences and turning them into a Hidden Markov model (HMM) using either PASTA or the slower, more accurate version of MAFFT. The HMM is used to incorporate all the remaining sequences one by one. The downside of this approach is that the seed sequences are insufficiently diverse and therefore preclude the accurate alignment of distantly related homologs to the seed HMM.

We considered that a regressive algorithm would address this problem by combining the benefits of a progressive approach when incorporating distant homologs with the improved accuracy of seeded methods. We needed to fulfill two simple constraints: the splitting of the sequences across sub-MSAs each containing a limited number of sequences and their combination into a MSA without the requirement of an alignment procedure. The main difference between our approach and existing ones lies in the order in which sequences are aligned, starting with the most diverse.

Given M sequences, the sub-MSAs are collected as follows. A clustering algorithm is first used to identify N nonoverlapping sequence groups of unspecified size—the children. N defines both the maximum number of children at any level and the maximum size of each sub-MSA. It constitutes the only free parameter of the algorithm. The first sub-MSA, the parent, is computed by selecting a representative sequence from each child group and by aligning theses N representatives with an MSA algorithm such as Clustal Omega (ClustalO), MAFFT or any suitable third-party software. The clustering algorithm is then re-applied onto every child group in which N new representatives are collected and multiply aligned to yield one child sub-MSA for each sequence in the parent. In each child group, the N new representatives are selected in such a way that the corresponding child MSA has exactly one sequence in common with its parent—the common representative. The procedure runs recursively by treating each child as a parent for the next generation until every sequence has been incorporated. The final MSA is produced by merging all the sub-MSAs. The merging of a child with its parent is done without additional alignment thanks to the common representative sequence. This sequence, present in both the child and its parent, enables the stacking of the corresponding positions (Fig. 1a). When doing so, insertions occurring within the representative, either in the child or in its parent, are projected as deletions (that is, gaps) in the other. Because of the way they are projected during merging, these insertions and their corresponding gap symbols do not need to be allocated in memory. They can be kept as counts and merely expanded while the MSA is written onto disk, thus dramatically decreasing the memory footprint.

A key step of this recursion is the clustering method and the subsequent selection of the N representative sequences. Our benchmarking suggests $N = 1{,}000$ to be a sensible choice (Supplementary Fig. 1a,b). This value is in agreement with a previous report on the largest number of sequences that can be directly aligned without accuracy loss[5]. The clusters were estimated from binary guide trees produced by existing large-scale MSA algorithms such as Clustal Omega (ClustalO) and MAFFT. The use of a binary tree to extract the most diverse sequences was inspired by an existing taxon sampling procedure[15]. In our implementation (Supplementary Note 1), every node gets labeled with the longest sequence among its

[1]Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain. [2]Institut de Biologia Evolutiva (UPF-CSIC), Universitat Pompeu Fabra, Barcelona, Spain. [3]Bioinformatics Studies, ESCI-UPF, Barcelona, Spain. [4]Institute of Science and Technology, Klosterneuburg, Austria. [5]Universitat Pompeu Fabra (UPF), Barcelona, Spain. *e-mail: evan.floden@crg.eu; cedric.notredame@crg.eu

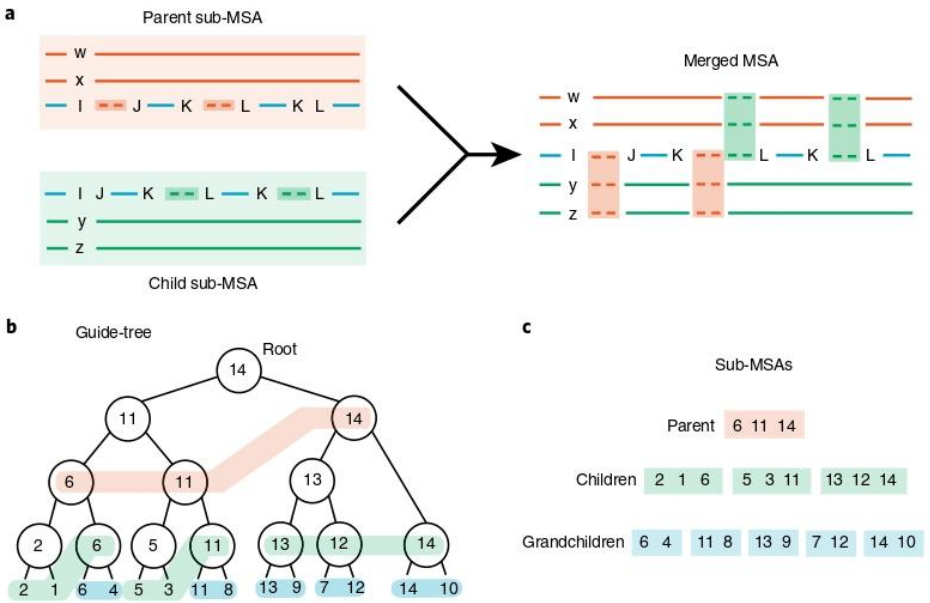


Fig. 1 | Regressive algorithm overview. a, Parent and children sub-MSAs are merged via their common sequence (blue) whose indels are projected from child to parent (green) and parent to child (red). **b**, The sub-MSAs are produced after collecting sequences from a binary guide tree with each node labeled with the name of its longest descendant sequence. Sequences are collected by traversing the tree in a breadth-first fashion. Pale red color blocks indicate how the *N* parent sequences (*N*=3) are collected by recursively expanding nodes. The same process is then applied to gather the children (green) and the grandchildren (blue). **c**, In the nine resulting sub-MSAs that are displayed, one should note the presence of a common representative sequence between each child and its parent.

Table 1 | TC score and average CPU time (s) on the 20 HomFam datasets containing over 10,000 sequences

		TC score (%)			CPU time (s)	
Tree method	**MSA algorithm**	**Nonregressive**	**Regressive**	**Reference**	**Nonregressive**	**Regressive**
PartTree	Fftns1	29.64	35.16	47.84	334	118
mBed	Fftns1	41.33	37.94	52.03	277	156
PartTree	ClustalO	26.94	42.21	50.54	3,017	377
mBed	ClustalO	39.03	41.91	53.71	570	338
Average		**34.24**	**39.31**	**51.03**	**1,050**	**247**
default/mBed	UPP	44.93	47.15	49.78	8,354	7,186
default/mBed	Sparsecore	44.98	51.06	53.50	2,313	3,184
PartTree	Gins1	-	47.54	49.46	-	12,478
mBed	Gins1	-	50.20	53.07	-	10,834

descendants. Given a fully labeled tree, the sequences of the first parent sub-MSA are collected by the breadth-first traversal of the tree, starting from the root through as many generations as required to collect *N* sequences (Fig. 1b). Because of the way they are collected along the tree, these *N* first sequences are as diverse as possible. Within the resulting sub-MSA every sequence is either a leaf or the representative of an internal node ready to be processed (Fig. 1c).

Our algorithm does not depend on specific alignment or guide-tree methods and therefore lends itself to be combined with any third-party software. This property enabled us to run various alignment software both directly and in combination with the regressive algorithm. A combination involves estimating a guide tree with an existing method, collecting sequences with the regressive algorithm and then computing the sub-MSAs with an existing MSA algorithm. By doing so we were able to precisely quantify the impact of our algorithm on both accuracy and computational requirements. We used as a benchmark the HomFam protein datasets[5] in which sequences with known structures—the references—are embedded among large numbers of homologs. Accuracy is estimated by aligning the large dataset and then comparing the induced alignment of the references with a structure-based alignment of these same references[16]. We started by benchmarking the ClustalO and MAFFT-FFTNS-1 (Fftns1) MSA algorithms using two guide-tree methods: ClustalO embedded *k*-means trees[17] (mBed) and MAFFT-PartTree[18] (PartTree). These widely adopted software packages were selected because they support large-scale datasets, are strictly progressive and allow the input and output of binary guide trees.

In three out of four combinations of guide tree and MSA algorithms, the regressive combination outperformed the progressive

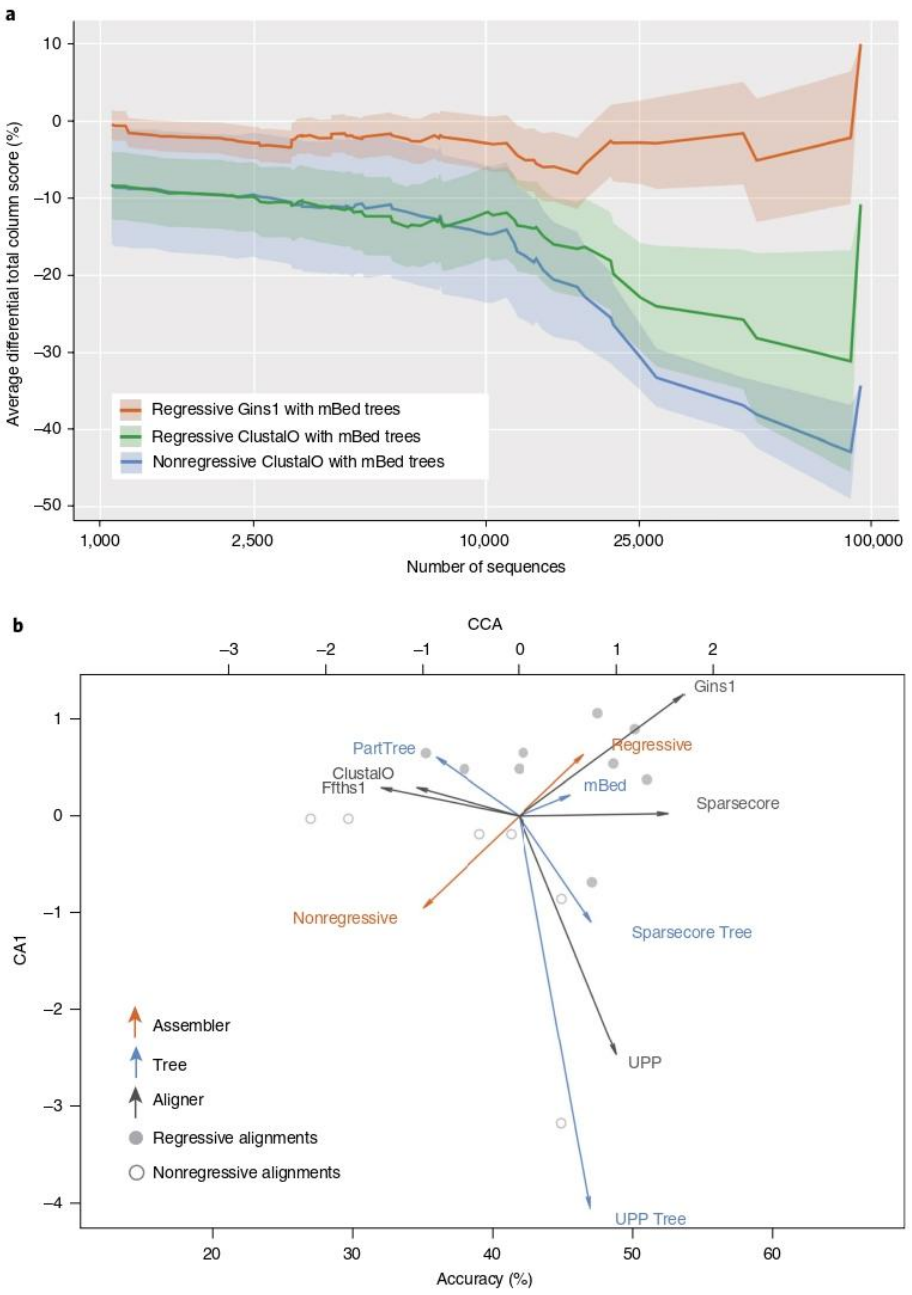


Fig. 2 | Relative performances of alternative MSA algorithm combinations. a, Average differential accuracy of datasets larger than number of sequences (horizontal axis). The differences of accuracy are measured between the reference sequence MSAs and their embedded projection in the large datasets. For each combination, $n=75$ independent MSA samples. The envelope is the standard deviation. **b**, In this CCA, the first component (horizontal axis, 14.1% of the variance) is constrained to be the TC score accuracy as measured on datasets larger than 10,000. The best unconstrained component (vertical axis) explains 20.8% of the remaining variance. Combinations (dots with their accuracy on the lower horizontal axis) are categorized by their guide tree (blue), MSA algorithms (gray) and regressive/nonregressive procedure (red). Vectors indicate the contributions to variance of each category from the three variables. Their projection onto the upper horizontal axis quantifies the contribution to variance of overall accuracy. For each combination, represented with a dot, $N=20$ independent MSA samples.

one. When considering the most discriminative measure (total column score (TC) in Table 1) on the datasets with over 10,000 sequences, the regressive combination delivered MSAs that were on average 5.13 percentage points more accurate than when computed progressively (39.31 and 34.24, respectively). These differences remained comparable, albeit reduced, when considering the contribution of smaller datasets (Supplementary Tables 1 and 2). Within this first set of analyses, the regressive combination of ClustalO with PartTree was the most accurate and on the large datasets it outperformed its progressive counterpart by 15.27 percentage points (42.21 and 26.94, respectively, Wilcoxon P value < 0.001).

We also tested the seed-based nonprogressive MSA algorithms Sparsecore[14] and UPP[13]. In both cases, their accuracy improved when combined with the regressive algorithm. For instance, the regressive combination of Sparsecore with mBed guide trees yielded the best readouts of this study on the very large alignments, and a clear improvement over the default Sparsecore (TC score 51.07 versus 44.98, Wilcoxon $P<0.1$). Comparable results were observed when extending this analysis to the sum-of-pair metrics or to smaller datasets (Supplementary Tables 1 and 2). The regressive algorithm is especially suitable for the scale-up of computationally expensive methods. For instance, the consistency-based variant of MAFFT named MAFFT-G-INS-1 (Gins1)[19], was among the most accurate small-scale MSA algorithms on the reference sequences. Gins1 cannot, however, be deployed on the HomFam datasets because its computational requirements are cubic with the number of sequences thus restricting it to a few hundred sequences. By combining Gins1 with the regressive algorithm we overcame this limitation and produced the most accurate readouts on datasets larger than 1,000 sequences (Supplementary Tables 1 and 2).

We complemented these measures of absolute accuracy with an estimate of accuracy degradation when scaling up. The effect of extra homologous sequences degrading the alignment accuracy of an MSA can be quantified by comparing the small MSAs of the reference sequences alone with their corresponding large-scale datasets. With the default progressive MSA algorithms ClustalO and Fftns1, the large datasets were on average 16.79 percentage points less accurate than when aligning the reference sequences on their own (Table 1, 34.24 and 51.03, respectively) with the trend being amplified on the larger alignments (Fig. 2a). Yet, on this same comparison, the regressive combinations were only affected by 11.72 points (Supplementary Fig. 2). The improved stability of the regressive combination was especially clear when considering Gins1 (Fig. 2a and Supplementary Fig. 2a) that was merely degraded by 2.87 percentage points thus achieving on the large datasets a level of accuracy close to the one measured on the reference sequences alone (Table 1, 50.20 and 53.07, respectively).

Identifying the factors driving accuracy improvement can be challenging considering that each alignment procedure relies on different combinations of algorithmic components (that is, regressive/nonregressive, tree method, MSA algorithm). For this purpose, we used constrained correspondence analysis (CCA)[20], a dimensionality reduction method adapted for categorical variables. When applied to Table 1 data, CCA allowed us to estimate the relative impact of each method's algorithmic component with respect to a constraining variable—accuracy in this case. As one would expect, the MSA algorithm is the most influential variable with respect to accuracy but CCA confirmed the general benefits of switching from a nonregressive to a regressive combination (Fig. 2b).

The most counterintuitive property of the regressive algorithm is its dependency on an initial parent MSA whose level of identity is imposed by the guide tree. Given an optimal guide tree, the level of identity of this initial parent is expected to be as low as possible. This first step is central to the algorithm's divide-and-conquer strategy, but it is unclear whether so much diversity at this early stage would harm accuracy prospects. We addressed this question by using HomFam to generate several alternative parent MSAs with different levels of identity for each dataset (that is, the same MSA algorithm and dataset but different guide trees). We then computed the final MSA corresponding to each parent and did not find any significant relationship between parent identity and final MSA accuracy (Supplementary Table 3). By contrast, a similar comparison across datasets (that is, same MSA algorithm and guide-tree method but different dataset) shows a strong positive dependency between parent identity and final MSA accuracy (Supplementary Table 4). This analysis confirms that, when using the regressive algorithm, the choice of very diverse sequences as a starting point does not incur a penalty while, as one would expect, datasets with lower identity result in MSAs with lower accuracy.

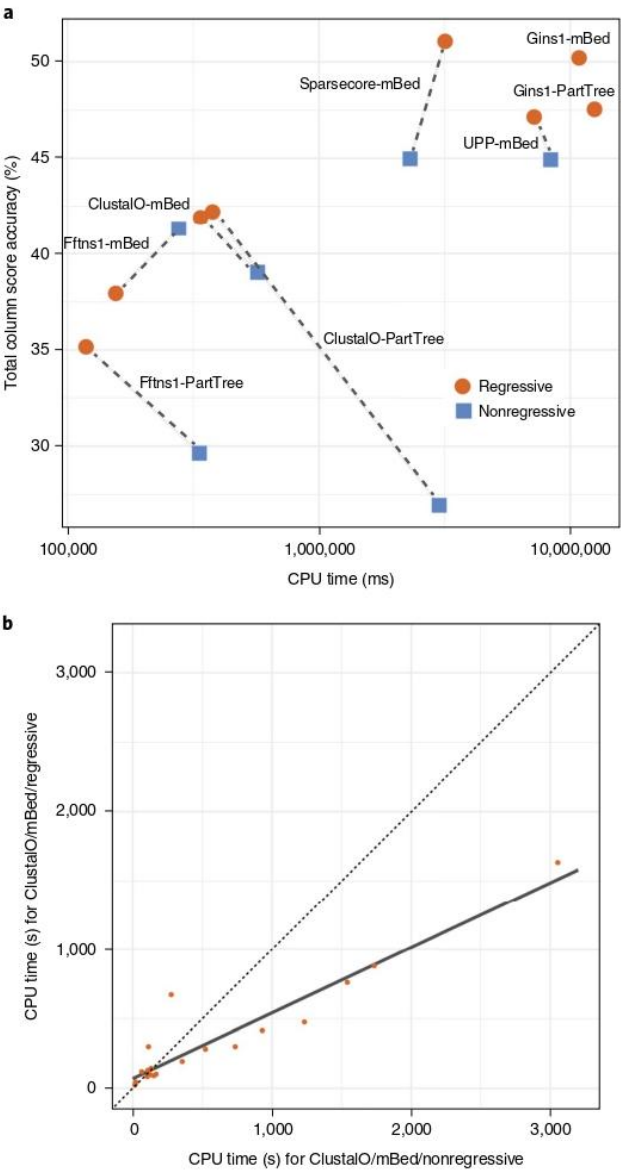


Fig. 3 | CPU requirements of the regressive algorithm on HomFam datasets containing more than 10,000 sequences. a, The total CPU requirements (horizontal axis) and average TC score accuracies (vertical axis). The corresponding nonregressive (blue square) and regressive (red circles) combinations are connected by a dashed line with the exception of Gins1 for which the nonregressive computation costs are prohibitive. For each combination, represented as a circles and squares, $N=20$ independent MSA samples. **b**, Comparison of CPU time requirements for ClustalO using mBed trees using a regressive and a nonregressive procedure on HomFam datasets containing more than 10,000 sequences. Each point represents an independent MSA. $N=20$ independent MSA samples. A linear regression (gray) was fitted on the resulting graph ($R^2=0.89$, $P=6.9\times10^{-10}$).

When using the same guide tree for the regressive and nonregressive alignment combinations, improved accuracy comes along with substantially improved computational performance. On average,

the regressive combinations require about fourfold less central processing unit (CPU) time than their nonregressive equivalent on datasets larger than 10,000 sequences (Table 1). Seeded methods such as UPP or Sparsecore appear to benefit less from the regressive deployment with marginal differences in CPU requirements (Fig. 3a). When considering MSA algorithms such as ClustalO or Fftns1 that scale linearly with the number of sequences, the improvement yielded by the regressive combination was roughly proportional to the original nonregressive CPU requirements. For instance, in the case of ClustalO using mBed trees, the regressive combination was about twice as fast as the progressive alignment and appeared to have a linear complexity (Fig. 3b). The situation was even more favorable when considering CPU intensive MSA algorithms such as Gins1 for which the nonregressive computation had been impossible.

We further explored the scaling up capacities of our algorithm using 45 Pfam 28.0 families[21] containing between 100,000 and 1.4 million sequences for the largest (ABC transporter family, PF00005). Although they lack a structural reference, these families were selected among the largest entries so as to provide a biologically realistic benchmark for scalability. When using a standard workstation (48 Gb of RAM, 160 CPU hours), the regressive methods were the only ones able to process all 45 datasets while the nonregressive methods tend to fail above 240,000 sequences and can only align a maximum of 500,000 sequences for the most robust (Supplementary Tables 5 and 6).

The ability to use slow and accurate MSA algorithms in linear time regardless of their original computational complexity is the most important feature of the regressive algorithm. It allows the application of any of these methods—natively—onto extremely large sequence datasets. This linearization is an inherent property of the regressive procedure in which all the sequences are split across sub-MSAs of a bounded size (that is, $N = 1{,}000$ sequences). This bounding in size results in a bounded computational cost. Since the total number of sub-MSAs is proportional to the initial number of sequences, the resulting complexity for the final MSA computation is linear. Furthermore, owing to the computational independence of the sub-MSAs, the regressive algorithm turns MSA computation into an embarrassingly parallel problem[22].

Our regressive algorithm provides a practical and generic solution to the critical problem of MSA scalability. It is a versatile algorithm that lends itself to further improvements—for instance, by exploring the impact of more sophisticated clustering structures, such as m-ary guide trees, or by testing different ways of selecting the representative sequences. The regressive algorithm is nonetheless a mature development framework that will enable a clean break between the improvement of highly accurate small-scale MSA algorithms—such as Gins1—and the design of more efficient large-scale clustering algorithms, such as PartTree and mBed. This divide will help potentiate the large body of work carried out in the clustering and alignment communities over the last decades and hopefully speed up the development of new improved methods. Achieving this goal is not optional. There is a Red Queen's race going on in genomics. It started the day omics' data growth overtook computing power and it shows no signs of slowing[23].

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41587-019-0333-6.

Received: 26 February 2019; Accepted: 29 October 2019;
Published online: 2 December 2019

References

1. Uguzzoni, G. et al. Large-scale identification of coevolution signals across homo-oligomeric protein interfaces by direct coupling analysis. *Proc. Natl Acad. Sci. USA* **114**, E2662–E2671 (2017).
2. Mirarab, S., Bayzid, M. S., Boussau, B. & Warnow, T. Statistical binning enables an accurate coalescent-based estimation of the avian tree. *Science* **346**, 1250463 (2014).
3. Wang, L. & Jiang, T. On the complexity of multiple sequence alignment. *J. Comput. Biol.* **1**, 337–348 (1994).
4. Hogeweg, P. & Hesper, B. The alignment of sets of sequences and the construction of phylogenetic trees. An integrated method. *J. Mol. Evol.* **20**, 175–186 (1984).
5. Sievers, F. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.* **7**, 539 (2011).
6. Lewin, H. A. et al. Earth BioGenome Project: sequencing life for the future of life. *Proc. Natl Acad. Sci. USA* **115**, 4325–4333 (2018).
7. Katoh, K., Misawa, K., Kuma, K. & Miyata, T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.* **30**, 3059–3066 (2002).
8. Chatzou, M. et al. Multiple sequence alignment modeling: methods and applications. *Brief. Bioinform.* **17**, 1009–1023 (2015).
9. Breen, M. S., Kemena, C., Vlasov, P. K., Notredame, C. & Kondrashov, F. A. Epistasis as the primary factor in molecular evolution. *Nature* **490**, 535–538 (2012).
10. Liu, K., Raghavan, S., Nelesen, S., Linder, C. R. & Warnow, T. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science* **324**, 1561–1564 (2009).
11. Mirarab, S. et al. PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. *J. Comput. Biol.* **22**, 377–386 (2015).
12. Collins, K. & Warnow, T. PASTA for proteins. *Bioinformatics* **34**, 3939–3941 (2018).
13. Nguyen, N.-P. D., Mirarab, S., Kumar, K. & Warnow, T. Ultra-large alignments using phylogeny-aware profiles. *Genome Biol.* **16**, 124 (2015).
14. Yamada, K. D., Tomii, K. & Katoh, K. Application of the MAFFT sequence alignment program to large data-reexamination of the usefulness of chained guide trees. *Bioinformatics* **32**, 3246–3251 (2016).
15. Minh, B. Q., Klaere, S. & von Haeseler, A. Phylogenetic diversity within seconds. *Syst. Biol.* **55**, 769–773 (2006).
16. Stebbings, L. A. & Mizuguchi, K. HOMSTRAD: recent developments of the homologous protein structure alignment database. *Nucleic Acids Res.* **32**, D203–D207 (2004).
17. Blackshields, G., Sievers, F., Shi, W., Wilm, A. & Higgins, D. G. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms Mol. Biol.* **5**, 21 (2010).
18. Katoh, K. & Toh, H. PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics* **23**, 372–374 (2007).
19. Katoh, K., Kuma, K.-I., Toh, H. & Miyata, T. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.* **33**, 511–518 (2005).
20. Greenacre, M. J. *Biplots in Practice* (Fundacion BBVA, 2010).
21. Finn, R. D. et al. Pfam: the protein families database. *Nucleic Acids Res.* **42**, D222–D230 (2013).
22. Herlihy, M. & Shavit, N. *The Art of Multiprocessor Programming* 1st edn (Morgan Kaufmann, 2012).
23. Kahn, S. D. On the future of genomic data. *Science* **331**, 728–729 (2011).

Methods

Reference datasets. The HomFam dataset was downloaded from the Clustal Omega website (http://www.clustal.org/omega/homfam-20110613-25.tar.gz). It features 94 families that contain homologous sequences extracted from Pfam 25. Each dataset is associated with a smaller set of reference sequences for which a structure-based alignment is available. For each family, the large-scale datasets are produced by merging the reference and the homologous sequences into a single file. The very large datasets were assembled by selecting 45 Pfam families whose sizes range between 100,000 and 1.4 million sequences. Summary statistics of the very large datasets are provided in Supplementary Table 7.

Multiple alignments and guide trees. The regressive algorithm is implemented in T-Coffee (hash cd5090c in the GitHub repository) and uses third-party methods for guide tree and sub-MSA computation. The sub-MSAs were produced using Clustal Omega (v.1.2.4), UPP (v.4.3.4) and MAFFT (v.7.397) for Gins1, Fftns1 and Sparsecore. The mBed and PartTree guide trees were estimated using the -guidetree-out option of Clustal Omega and the -parttree option of MAFFT. Random trees were generated by shuffling the taxa on the original mBed trees (+newick_randomize option in T-Coffee/seq_reformat). Parent MSAs were collected using a specific T-Coffee flag triggering their output as intermediate files (DUMP_ALN_BUCKET = 1).

Benchmarking. Benchmarking was carried out by aligning either the reference or the large-scale datasets, and by comparing the projection of the reference sequences with their reference alignment using the aln_compare option of the T-Coffee package. This option supports the sum-of-pairs (fraction of pairs of residues in the reference alignment found in the benchmark) and the TC score (fraction of columns in the reference alignment found in the benchmark) metrics[24].

Constrained correspondence analysis. Each alignment procedure (for example, regressive ClustalO using mBed tTrees) is represented in the form of a string of zeros and ones encoding its categorical variables (guide-tree method, aligner, assembler). Within this string, each variable is encoded in a substring whose length is equal to the number of levels (for example, number of alternative guide-tree methods). These substrings therefore contain a single entry so that the entire string sums to the number of variables for any given procedure (that is, three in our case). Once encoded this way alignment procedures become the rows of an indicator matrix that can be analyzed with dimensionality reduction techniques such as multiple correspondence analysis. In constrained correspondence analysis (CCA; also known as canonical correspondence analysis) dimensionality reduction is guided by additional information about each observation. In our case, this information is the accuracy (TC score) of each alignment procedure averaged across the 20 datasets containing over 10,000 sequences. The application of CCA involves projecting the indicator matrix onto a linear space defined by the accuracy vector[20]. The technique makes it possible to then perform a singular value decomposition and displayed in the form of a biplot as in Fig. 2b. Calculations were carried out using the R package Vegan (https://cran.r-project.org/package=vegan). Percentage variance explained is obtained by dividing the eigenvalue of the respective axis with the sum of all the eigenvalues, multiplied by 100.

Relationship between parent MSA identity and accuracy. The 75 HomFam datasets containing more than 1,000 sequences were regressively aligned using three guide-tree methods (mBed, PartTree and randomized mBed) along with four MSA algorithms (ClustalO, Fftns1, UPP and Sparsecore). For a given dataset, the use of different guide trees usually results in different parent MSAs. We therefore collected all 900 pairs of combinations involving the same dataset, the same MSA algorithms and two different guide trees. Results were compiled in a contingency table counting increase or decrease of the parent MSA percent identity as well as increase or decrease of the MSA accuracy (as measured by TC score). A two-sided Fisher test (implemented in R) was used to test the null hypothesis of no association (that is, odds ratio 1). The ratio of the odds of increasing accuracy versus decreasing accuracy was 0.85 times higher for the cases where identity increased than for the cases where identity decreased ($P < 0.29$). To do a comparable analysis across different datasets, we collected all the 33,000 pairs of combinations involving a different dataset, the same MSA algorithms and the same guide trees. Here, the odds of increasing versus decreasing accuracy were 4.1 times higher in the cases where identity increased than in the cases where identity decreased ($P < 10^{-15}$).

Computation. All computation was carried out on a cluster running Scientific Linux release 7.2 with all guide trees, alignments and evaluations carried out within a container based on the Debian (Jessie) operating system. The computational pipeline (see the Code availability section) was implemented in the Nextflow language[25] and was deployed in a containerized form using Singularity. Computation was limited to 48 Gb of memory and 160 CPU hours. Given a HomFam family, this pipeline generates the mBed and PartTree guide trees for both the reference and the large-scale dataset. It then combines the selected aligners (ClustalO, Gins1, Fftns1, Sparsecore and UPP) and the precomputed guide trees to generate (1) a default alignment of the reference sequences (2) a default (that is, nonregressive) alignment of the large-scale dataset and (3) a regressive alignment of the large-scale dataset. Note that UPP and Sparsecore do not support external guide trees and that their default alignments were therefore produced using the default guide-tree procedures of these methods. A Docker image has been created that contains all the pipeline dependencies. It is available from DockerHub (https://hub.docker.com) and is available via the following command:

```
docker pull cbcrg/regressive-msa:cd5090c
```

The Dockerfile is also provided in the Git repository to allow for reuse and addition of new tools. All command lines used by the pipeline are also provided in the dedicated Supplementary Materials section (Supplementary Note 2).

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

All data, analyses and results are available from Zenodo (https://doi.org/10.5281/zenodo.3271452).

Code availability

The regressive alignment algorithm has been implemented in T-Coffee and is available at the T-Coffee website (http://www.tcoffee.org) and on GitHub (https://github.com/cbcrg/tcoffee). A GitHub repository containing the Nextflow workflow[25] and Jupyter notebooks[26] to replicate the analysis are available at https://github.com/cbcrg/dpa-analysis (release v.1.2).

References

24. Thompson, J. D., Plewniak, F. & Poch, O. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics* **15**, 87–88 (1999).
25. Di Tommaso, P. et al. Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
26. Perkel, J. M. Why Jupyter is data scientists' computational notebook of choice. *Nature* **563**, 145–146 (2018).

Acknowledgements

We thank G. Riddihough for revisions and comments on the manuscript and O. Gascuel for suggestions. This project was supported by the Centre for Genomic Regulation, the Spanish Plan Nacional, the Spanish Ministry of Economy and Competitiveness, 'Centro de Excelencia Severo Ochoa' (E.G., P.T., C.M., I.E., L.M., A.B., F.K., E.F. and C.N.) and an ERC Consolidator Grant from the European Commission, grant agreement no. 771209 ChrFL (F.K.).

Author contributions

C.N. designed and implemented the algorithm. E.F., E.G., L.M., A.B. and P.D.T designed the validation procedure and carried out the validation. I.E. performed statistical and CCA analyses. E.F., C.N., E.G., C.M., L.M., A.B., P.D.T., I.E., F.K. and H.L. wrote and edited the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41587-019-0333-6.

Correspondence and requests for materials should be addressed to E.F. or C.N.

Reprints and permissions information is available at www.nature.com/reprints.

2.2: Supplementary Information

Supplementary Information for "Large multiple sequence alignments with a root-to-leaf regressive method"

Supplementary Notes

Supplementary Note 1: Regressive algorithm pseudocode. The exact implementation is available on the GitHub repository as part of the T-Coffee latest distribution (https://github.com/cbcrg/tcoffee). The following pseudo-code provides a simplified overview of a serial version of the algorithm assuming a binary tree in which every node is labelled with the name of the longest sequence among its descendant leaves. The symbol node refers to three-ways nodes in this binary tree where each node has a parent, a left child, a right child and a label, with the exception of leaves that have no child and the root that has no parent. The regressive algorithm traverses the binary tree in a recursive fashion, when doing so, it collects N sequences, multiply align them and merges the resulting sub-MSAs. The code is initiated with the node=root of the labelled tree.

```
node2subMSA (node)
    // expands node into a linked list
    // start->node1->...->nodeN->end
    // Turns the linked list into a parent_MSA
    // recursively expand each parent_MSA sequence into a child_MSA
    // merges every child_MSA within the parent_MSA
    // returns the parent_MSA
    // gets initialized with node2subMSA (root)

    if (node is leaf) return node->seq

    start = declare_node()    //empty start node
    end  = declare node()     //empty end node

    //define the linked list: start->node->end
    node->next = end; end-> prev = node
    node->prev = start; start-> next = node

    n=1           //sequences counter
    nleaf=0       //leaf counter

    //Expand all the nodes, one generation at a time

    while (n<N and !(node==end and n==nleaves))

      //rewind if all the nodes if
      //the current generation has been expanded
      if (node==end)node=start->next

      //replace current node with its left and right children
      //Turn pnode->node->nnode into pnode->left->right->nnode
      if (node is not leaf)
            pnode=node->previous
            nnode=node->next

            pnode->next=node->left
            node->left->prev=pnode
            node->left->next=node->right
            node->right->prev=node->left
```

```
            node->left->next=nnode
            nnode->prev=node->left

            //the sequence list is now one sequence longer
            n=n+1

            //keep advancing one node at a time
            node=nnode

      else
            //keep track of the leaf nodes
            nleaves=nleaves+1

    //Collect all the sequences from start->node1->node2…->end
    node=start->next
    while (node !=end)
      add node->label to SeqList
      node=node->next

    //Send collected sequences to a third party aligner
    parent_MSA=MSA_Algorithm(SeqList)

    //Treat each internal node in parent_MSA as a start node
    //collect each child_MSA and merge it with parent_MSA
    node=start->next
    while (node!=end)
      child_MSA=node2subMSA (node)
      parent_MSA=merge_subMSAs (node->seq, parent_MSA, child_MSA)
      node=node->next

    //parent_MSA contains ALL descendant sequences of node
    //if node is the root, parent_MSA is the final MSA
    return parent_MSA

MSA_Algorithm (SeqList)
    subMSA=system (Run third party MSA Algorithm on SeqList)
    return subMSA

merge_subMSAs (seq, parent_MSA,child_MSA)
    //seq is the representative sequence
    //seq is present in parent_MSA and in child_MSA

    foreach residues i of seq
      parent_gap=number of '-' between i and i+1 in parent_MSA
      child_gap=number of '-' between i and i+1 in child_MSA

      columnP=parent_MSA column containing residue i of seq
      for each sequence in parent_MSA
            Insert child_gap '-' after columnP

      columnC=child_MSA column containing residue i of seq
      for each sequence in child_MSA
            Insert parent_gap '-' after columnC

    //parent_MSA and Child_MSA now have the same length

    Replace seq in parent_MSA with all sequences of child_MSA
    Return parent_MSA
```

Supplementary Note 2: Command Lines. The following section lists all the command lines used for data generation.

Workflow deployment. In order to carry out a reproducible deployment of guide tree and alignments, the Nextflow workflow manager was used to deploy containerized versions of the software using Docker or Singularity. Procedures described in the following section can be reproduced by downloading the Nextflow software (*www.nextflow.io*) and running the following command on the Homfam dataset:

```
nextflow run cbcrg/dpa-analysis --seqs="./data/seqs/*.fa" --refs="./data/refs/*.ref' -
with-singularity
```

The individual commands within the workflow are described below.

Large scale tree generation.
mBed and PartTree trees were generated as follows:

mBed:
```
clustalo -i ${seqs} --guidetree-out ${id}.${tree_method}.dnd --force
```

PartTree:
```
t_coffee -other_pg seq_reformat -in ${seqs} -action +seq2dnd parttree -output newick >>
     ${id}.${tree_method}.dnd
```

This procedure is a wrapper for the MAFFT-PartTree command that produces a coded binary tree, that must be recoded into a standard newick tree

mBed Random:
```
clustalo -i ${seqs} --guidetree-out ${id}.CLUSTALO.dnd

t_coffee -other_pg seq_reformat -in ${id}.CLUSTALO.dnd -action +newick_randomize 1 >>
${id}.${tree_method}.dnd

rm ${id}.CLUSTALO.dnd
```

Non-regressive MSA generation.
The following commands were used to generate non-regressive MSAs.

Clustal Omega:
```
clustalo --infile=${seqs}  --guidetree-in=${guide_tree}  --outfmt=fa
     –o ${id}.std.${align_method}.with.${tree_method}.tree.aln
```

MAFFT-FFT-NS-1:
```
t_coffee -other_pg seq_reformat -in ${guide_tree} -input newick -in2 ${seqs} -input2
fasta_seq -action +newick2mafftnewick >> ${id}.mafftnewick


newick2mafft.rb 1.0 ${id}.mafftnewick > ${id}.mafftbinary

/mafft/bin/mafft --retree 1 --anysymbol --treein ${id}.mafftbinary ${seqs} >
     ${id}.std.${align_method}.with.${tree_method}.tree.aln
```

MAFFT-G-INS-1:
```
t_coffee -other_pg seq_reformat -in ${guide_tree} -input newick -in2 ${seqs} -input2
fasta_seq -action
     +newick2mafftnewick >> ${id}.mafftnewick

newick2mafft.rb 1.0 ${id}.mafftnewick > ${id}.mafftbinary

ginsi --treein ${id}.mafftbinary ${seqs}
>${id}.std.${align_method}.with.${tree_method}.tree.aln
```

MAFFT-Sparsecore:

```
replace_U.pl ${seqs} // This is a Mafft pre-processing routine

/mafft/bin/mafft-sparsecore.rb -i ${seqs} > ${id}.default.${align_method}.aln
```

UPP:

```
replace_U.pl ${seqs}

run_upp.py -s ${seqs}  -m amino -x 1 -o ${id}.default.${align_method}

mv ${id}.default.${align_method}_alignment.fasta ${id}.default.${align_method}.aln
```

Regressive MSA generation.

In the following command lines, the flag -dpa_method refers to the alignment method deployed by the T-Coffee. The exact command lines corresponding to these modes are available from the file *lib/util_lib/util_constraints_list.c* in the T-Coffee distribution and are identical to the ones used for the non-regressive method in the previous section.

Regressive Clustal Omega:

```
t_coffee -dpa -dpa_method clustalo_msa   -dpa_tree ${guide_tree} -seq ${seqs}  -dpa_nseq
${bucket_size} –outfile
${id}.dpa_${bucket_size}.${align_method}.with.${tree_method}.tree.aln
```

Regressive MAFFT-FFT-NS-1:

```
t_coffee -dpa -dpa_method mafftfftns1_msa -dpa_tree ${guide_tree} -seq ${seqs} -dpa_nseq
${bucket_size} –outfile
${id}.dpa_${bucket_size}.${align_method}.with.${tree_method}.tree.aln
```

Regressive MAFFT-G-INS-1:

```
t_coffee -dpa -dpa_method mafftginsi_msa -dpa_tree ${guide_tree} -seq ${seqs} -dpa_nseq
${bucket_size} –outfile
${id}.dpa_${bucket_size}.${align_method}.with.${tree_method}.tree.aln
```

Regressive MAFFT-Sparsecore:

```
t_coffee -dpa -dpa_method mafftsparsecore_msa -dpa_tree ${guide_tree} -seq ${seqs} -
dpa_nseq
${bucket_size} -outfile
${id}.dpa_${bucket_size}.${align_method}.with.${tree_method}.tree.aln
```

Regressive UPP:

```
t_coffee -dpa -dpa_method upp_msa -dpa_tree ${guide_tree} -seq ${seqs} -dpa_nseq
${bucket_size} -outfile
${id}.dpa_${bucket_size}.${align_method}.with.${tree_method}.tree.aln
```

Benchmarking.

The commands used for the MSA evaluation using the metrics implemented with the T-Coffee package using *aln_compare* were executed as follows:

Sum Of Pairs (SoP):

```
t_coffee -other_pg aln_compare -al1 ${ref_alignment} -al2 ${test_alignment}.compare_mode
sp | grep -v "seq1" |grep -v '*' | awk '{ print \$4}' ORS="\t" >
"${id}.${align_type}.${bucket_size}.${align_method}.${tree_method}.sp"
```

Total Column Score (TC):

```
t_coffee -other_pg aln_compare -al1 ${ref_alignment} -al2 ${test_alignment} -
compare_mode tc | grep -v "seq1" |grep -v '*' | awk '{ print \$4}' ORS="\t" >
"${id}.${align_type}.${bucket_size}.${align_method}.${tree_method}.tc"
```

Supplementary Tables

Supplementary Table 1. Summary of SoP and TC values collected over HomFam datasets of various sizes.

A		Over 10.000 sequences - 20 Datasets							
		Sums of Pairs (SoP)			Total Column Score (TC)			CPU (s)	
Tree Method	MSA Algorithm	Non regressive	Regressive	Reference	Non regressive	Regressive	Reference	Non regressive	Regressive
		Score (%)	Score (%)	Score (%)	Score (%)	Score (%)	Score (%)		
PartTree	Fftns1	56.34	59.74	77.10	29.64	35.16	47.84	334	118
mBed	Fftns1	68.71	64.00	78.20	41.33	37.94	52.03	277	155
PartTree	ClustalO	50.13	66.58	78.14	26.94	42.21	50.54	3,016	377
mBed	ClustalO	64.97	71.11	80.54	39.03	41.91	53.71	570	338
Average		**60.04**	**65.36**	**78.50**	**34.24**	**39.31**	**51.03**	**1,049**	**247**
default/mBed	UPP	70.95	72.39	80.27	43.80	47.15	49.88	8,353	7,186
default/mBed	Sparsecore	72.31	77.28	81.67	44.98	51.06	53.51	2,313	3,184
PartTree	Gins1	-	71.51	78.20	-	47.54	49.46	-	12,477
mBed	Gins1	-	77.14	81.66	-	50.20	53.07	-	10,833

B		Between 1,000 and 10,000 sequences - 55 Datasets							
		Sums of Pairs (SoP)			Total Column Score (TC)			CPU (s)	
Tree Method	MSA Algorithm	Non regressive	Regressive	Reference	Non regressive	Regressive	Reference	Non regressive	Regressive
		Score (%)	Score (%)	Score (%)	Score (%)	Score (%)	Score (%)		
PartTree	Fftns1	72.89	72.51	84.41	44.81	47.43	60.33	9	17
mBed	Fftns1	80.77	78.29	85.93	56.38	53.75	63.73	7	28
PartTree	ClustalO	74.44	78.25	87.25	50.50	55.15	66.08	367	77
mBed	ClustalO	83.15	83.17	87.91	61.45	60.33	67.62	138	82
Average		**77.81**	**78.06**	**86.38**	**53.29**	**54.16**	**64.44**	**130**	**51**
default/mBed	UPP	80.13	80.83	85.67	59.01	57.34	62.79	2,380	1,740
default/mBed	Sparsecore	85.24	86.84	87.69	60.10	65.10	65.86	633	1,091
PartTree	Gins1	-	83.53	86.87	-	61.04	64.15	-	3,974
mBed	Gins1	-	87.44	87.53	-	66.00	65.57	-	3,878

C		Over 92 sequences - 94 Datasets							
		Sums of Pairs (SoP)			Total Column Score (TC)			CPU (s)	
Tree Method	MSA Algorithm	Non regressive	Regressive	Reference	Non regressive	Regressive	Reference	Non regressive	Regressive
		Score (%)	Score (%)	Score (%)	Score (%)	Score (%)	Score (%)		
PartTree	Fftns1	70.95	71.71	83.64	44.11	47.13	59.43	76	35
mBed	Fftns1	78.46	75.99	84.94	53.89	51.42	62.82	63	49
PartTree	ClustalO	71.28	77.89	85.77	47.81	55.24	63.66	858	126
mBed	ClustalO	80.41	81.73	86.67	58.25	58.21	65.19	203	121
Average		**75.28**	**76.83**	**85.26**	**51.02**	**53.00**	**62.78**	**300**	**83**
default/mBed	UPP	78.44	79.25	85.04	55.36	55.50	61.06	3,211	2,590
default/mBed	Sparsecore	82.86	84.92	86.77	57.81	62.14	64.06	880	1,333
PartTree	Gins1	-	81.66	85.21	-	58.89	61.43	-	5,021
mBed	Gins1	-	85.14	86.68	-	62.36	63.68	-	4,579

(**A**) Scores measured on the 20 HomFam datasets containing more than 10,000 sequences. On each line, an Alignment algorithm is deployed in non-regressive and regressive mode using a Tree Method. Reference indicates the score of the seed alignment using the non-regressive method (note that seed alignments contain less than 10 sequences and are identically aligned by regressive and non-regressive protocols). The Central Processing Unit Time column (CPU) indicates the amount of time (seconds) used by the regressive and non-regressive protocols respectively averaged by the number of datasets. Data displayed for TC score is the same as used for Table 1 in the main text. In the SoP and TC sections, the best scoring regressive readout is highlighted in yellow and the best non-regressive is framed in red. The same layout is used on (**B**) where measures relate to the 55 datasets having between 1,000 and 10,000 sequences and (**C**) for all the 94 HomFam datasets containing 92 sequences or more.

Supplementary Table 2. Wilcoxon signed-rank test p-values for differences between the regressive and non-regressive readouts.

A)

Over 10,000 Sequences - TC

Non Regressive / Regressive	MSA Algorithm	ClustalO	ClustalO	Fftns1	Fftns1	Sparsecore	UPP
MSA Algorithm	Tree method	mBed	PartTree	mBed	PartTree	mBed	mBed
ClustalO	mBed	9.57E-02	7.75E-05	6.11E-01	3.11E-03	8.78E-01	9.01E-01
ClustalO	PartTree	1.31E-01	1.09E-04	2.37E-01	2.79E-03	8.35E-01	8.01E-01
Fftns1	mBed	3.64E-01	1.58E-03	8.92E-01	1.33E-02	9.91E-01	9.53E-01
Fftns1	PartTree	5.94E-01	9.93E-04	9.46E-01	6.15E-02	9.90E-01	9.78E-01
Gins1	mBed	2.14E-03	8.39E-05	1.45E-03	9.08E-05	7.85E-02	9.20E-02
Gins1	PartTree	6.04E-03	8.56E-04	8.59E-03	5.25E-05	3.74E-01	2.37E-01
Sparsecore	mBed	2.75E-03	8.39E-05	2.84E-03	6.59E-05	8.51E-02	6.31E-02
UPP	mBed	2.21E-02	6.05E-04	7.95E-02	4.19E-04	7.53E-01	5.09E-01

B)

Over 10,000 Sequences - SoP

Non Regressive / Regressive	MSA Algorithm	ClustalO	ClustalO	Fftns1	Fftns1	Sparsecore	UPP
MSA Algorithm	Tree method	mBed	PartTree	mBed	PartTree	mBed	mBed
ClustalO	mBed	2.00E-02	5.57E-05	2.86E-01	6.87E-04	8.86E-01	9.17E-01
ClustalO	PartTree	2.22E-01	4.78E-05	8.00E-01	1.91E-02	9.44E-01	9.25E-01
Fftns1	mBed	3.07E-01	1.69E-03	9.52E-01	1.20E-02	9.97E-01	9.87E-01
Fftns1	PartTree	7.19E-01	2.15E-03	9.77E-01	1.80E-01	9.98E-01	9.87E-01
Gins1	mBed	2.01E-03	3.18E-04	9.08E-04	7.15E-05	3.35E-02	2.32E-02
Gins1	PartTree	2.73E-02	9.13E-04	1.90E-01	2.24E-04	5.52E-01	4.48E-01
Sparsecore	mBed	2.92E-03	2.09E-04	2.01E-03	7.15E-05	6.56E-02	1.97E-02
UPP	mBed	4.95E-02	9.13E-04	1.83E-01	1.58E-04	8.38E-01	5.95E-01

C)

Between 1,000 and 10,000 Sequences - TC

Non Regressive / Regressive	MSA Algorithm	ClustalO	ClustalO	Fftns1	Fftns1	Sparsecore	UPP
MSA Algorithm	Tree method	mBed	PartTree	mBed	PartTree	mBed	mBed
ClustalO	mBed	6.23E-01	2.51E-08	1.11E-02	3.03E-08	8.93E-01	3.27E-01
ClustalO	PartTree	9.95E-01	7.56E-05	3.50E-01	1.55E-04	9.95E-01	8.51E-01
Fftns1	mBed	9.99E-01	1.25E-02	6.95E-01	7.68E-06	1.00E+00	9.31E-01
Fftns1	PartTree	1.00E+00	7.32E-01	9.97E-01	8.92E-02	1.00E+00	9.96E-01
Gins1	mBed	6.03E-04	6.84E-08	1.31E-07	3.22E-10	8.04E-02	1.53E-04
Gins1	PartTree	2.08E-01	5.25E-07	1.28E-03	8.25E-08	5.23E-01	7.36E-02
Sparsecore	mBed	2.64E-03	2.23E-07	6.37E-06	8.80E-10	9.36E-02	2.65E-03
UPP	mBed	6.27E-01	3.19E-03	8.75E-02	3.64E-05	9.30E-01	2.07E-01

D)

Between 1,000 and 10,000 Sequences - SoP

Non Regressive / Regressive	MSA Algorithm	ClustalO	ClustalO	Fftns1	Fftns1	Sparsecore	UPP
MSA Algorithm	Tree method	mBed	PartTree	mBed	PartTree	mBed	mBed
ClustalO	mBed	4.35E-01	5.01E-09	6.84E-03	5.28E-09	9.94E-01	2.02E-01
ClustalO	PartTree	9.93E-01	2.90E-05	4.59E-01	8.15E-04	1.00E+00	8.13E-01
Fftns1	mBed	9.99E-01	1.64E-03	7.96E-01	6.47E-06	1.00E+00	9.06E-01
Fftns1	PartTree	1.00E+00	6.87E-01	9.96E-01	2.14E-01	1.00E+00	9.90E-01
Gins1	mBed	1.59E-05	6.54E-09	4.51E-08	8.83E-11	2.58E-01	7.17E-05
Gins1	PartTree	5.85E-02	2.34E-08	1.70E-03	2.23E-07	7.81E-01	5.38E-02
Sparsecore	mBed	1.12E-04	3.83E-08	1.27E-06	9.86E-11	3.16E-01	9.54E-04
UPP	mBed	3.67E-01	6.64E-05	1.04E-01	8.63E-04	9.97E-01	2.03E-01

E)

Over 92 Sequences - TC

Non Regressive / Regressive	MSA Algorithm	ClustalO	ClustalO	Fftns1	Fftns1	Sparsecore	UPP
MSA Algorithm	Tree method	mBed	PartTree	mBed	PartTree	mBed	mBed
ClustalO	mBed	3.62E-01	3.82E-14	8.85E-03	1.69E-11	9.72E-01	4.93E-01
ClustalO	PartTree	9.44E-01	3.16E-10	1.01E-01	8.12E-08	9.96E-01	7.52E-01
Fftns1	mBed	9.98E-01	1.64E-04	8.88E-01	3.98E-08	1.00E+00	9.92E-01
Fftns1	PartTree	1.00E+00	6.04E-02	9.99E-01	2.68E-03	1.00E+00	9.99E-01
Gins1	mBed	1.71E-05	8.34E-11	2.74E-11	6.79E-15	4.07E-02	5.76E-05
Gins1	PartTree	1.88E-02	7.86E-10	4.28E-06	4.74E-12	5.10E-01	2.37E-02
Sparsecore	mBed	6.32E-05	1.67E-10	1.79E-09	1.38E-14	1.73E-02	2.01E-04
UPP	mBed	3.88E-01	4.83E-05	2.46E-02	8.83E-08	9.89E-01	3.18E-01

F)

Over 92 Sequences - SoP

Non Regressive / Regressive	MSA Algorithm	ClustalO	ClustalO	Fftns1	Fftns1	Sparsecore	UPP
MSA Algorithm	Tree method	mBed	PartTree	mBed	PartTree	mBed	mBed
ClustalO	mBed	2.00E-02	5.57E-05	2.86E-01	6.87E-04	8.86E-01	9.17E-01
ClustalO	PartTree	2.22E-01	4.78E-05	8.00E-01	1.91E-02	9.44E-01	9.25E-01
Fftns1	mBed	3.07E-01	1.69E-03	9.52E-01	1.20E-02	9.97E-01	9.87E-01
Fftns1	PartTree	7.19E-01	2.15E-03	9.77E-01	1.80E-01	9.98E-01	9.87E-01
Gins1	mBed	2.01E-03	3.18E-04	9.08E-04	7.15E-05	3.35E-02	2.32E-02
Gins1	PartTree	2.73E-02	9.13E-04	1.90E-01	2.24E-04	5.52E-01	4.48E-01
Sparsecore	mBed	2.92E-03	2.09E-04	2.01E-03	7.15E-05	6.56E-02	1.97E-02
UPP	mBed	4.95E-02	9.13E-04	1.83E-01	1.58E-04	8.38E-01	5.95E-01

(**A**) The p-value estimates the probability of the regressive and non-regressive having the same accuracy distribution while using the alternative hypothesis of systematically higher accuracies in the regressive mode (one-sided Wilcoxon signed-rank test, n=20 independent MSA samples). The test was made on datasets larger than 10,000 and using the Total Column score as a readout. Non-significant cells are colored in dark red (0.1<P<1), low significance cells are in light green (0.01<P<0.1), green (0.001<P<0.01) and dark green (P<0.001). The cells with bold borders contain p-values of the difference between directly comparable protocols (i.e. analysis displayed in the same line in Table 1). Similar analyses were made using SoP as a readout (**B**), TC and SoP for datasets containing between 1,000 and 10,000 sequences (**C, D**) respectively (n=55 independent MSA samples), and SoP and TC on all the 92 HomFam datasets (**E**,**F**)(n=94 independent MSA samples).

Supplementary Table 3. Contingency table summarizing the accuracy impact of identity variation in the parent MSA across alternative MSAs of the same dataset.

Contingency Table Across Replicates – Same MSA algorithm, Different Tree Method, Same Dataset (%)				
	Delta TC>0 (%)	**Delta TC=0 (%)**	**Delta TC<0 (%)**	
Delta ID>0 (%)	30.44	2.33	16.56	
Delta ID=0 (%)	6.44	0	2.44	
Delta ID<0 (%)	27.78	1.22	12.78	
SUM: 44.33				SUM: 43.22
Total Counts	900			

Pairs of alternative alignments of the same dataset carried out with the same aligner but different types of guide trees. On each MSA the accuracy was measured (TC) as well as the identity level of the parent MSA. These accuracies and identities were then compared (Delta TC and Delta ID) and the resulting variations were counted so as to fill up the matrix. The entries highlighted in red and green correspond to the cases in which the variation in identity and accuracy follow the same trend. The bottom left and right values provide the sum of these entries respectively.

Supplementary Table 4. Contingency table summarizing the accuracy impact of identity variation in the parent MSA across different datasets.

Contingency Table Across Datasets – Same MSA Algorithm, Same Tree Method, Different Dataset (%)				
	Delta TC>0 (%)	**Delta TC=0 (%)**	**Delta TC<0 (%)**	
Delta ID>0 (%)	46.26	0.09	12.56	
Delta ID=0 (%)	2.01	0	1.29	
Delta ID<0 (%)	17.86	0.07	19.87	
SUM: 30.42				SUM: 66.13
Total Count	33300			

Pairs of MSA of different datasets aligned regressively with the same MSA algorithm and the same guide tree method. On each MSA the accuracy was measured (TC) as well as the identity level of the parent MSA. These accuracies and identities were then compared (Delta TC and Delta ID) and the resulting variations were counted so as to fill up the matrix. The entries highlighted in red and green correspond to the cases in which the variation in identity and accuracy follow the same trend. The bottom left and right values provide the sum of these entries respectively.

Supplementary Table 5. Maximum number of sequences aligned before failure.

Maximum number of sequences aligned before failure			
MSA Algorithm	***Tree Method***	***Non-Regressive***	***Regressive***
ClustalO	mBed	231,015	547,587
ClustalO	PartTree	111,802	**1,466,247**
Fftns1	mBed	239,488	547,587
Fftns1	PartTree	231,015	**1,466,247**
UPP	Internal	102,112	-
UPP	mBed	-	547,587
UPP	PartTree	-	**1,466,247**
Sparsecore	Internal	102,112	-
Sparsecore	mBed	-	547,587
Sparsecore	PartTree	-	547,587

The 45 Pfam datasets were aligned while allowing for a maximum of 48 Gb RAM and 160 CPU hours. Instances were sorted by number of input sequences and the size of the first one before failure was recorded. The largest value in this table corresponds to the largest instance and no failure was recorded in the cases when this instance was computed.

Supplementary Table 6. Maximum number of sequences aligned.

Maximum number of sequences aligned			
MSA Algorithm	***Tree Method***	***Non-Regressive***	***Regressive***
ClustalO	mBed	547,587	547,587
ClustalO	PartTree	328,784	**1,466,247**
Fftns1	mBed	328,784	547,587
Fftns1	PartTree	328,784	**1,466,247**
UPP	Internal	547,587	-
UPP	mBed	-	547,587
UPP	PartTree	-	**1,466,247**
Sparsecore	Internal	239,488	-
Sparsecore	mBed	-	547,587
Sparsecore	PartTree	-	547,587

The 45 Pfam datasets were aligned while allowing for a maximum of 48 Gb RAM and 160 CPU hours. Instances were sorted by number of input sequences and the size of the largest instance computed was recorded. The largest value in this table (1,466,247) corresponds to the largest instance and no failure was recorded in the cases when this instance was computed.

Supplementary Table 7. Summary Statistics of the 45 Pfam very large datasets.

Pfam 28.0 Family	Number of sequences	Number of residues		Pfam 28.0 Family	Number of sequences	Number of residues
PF00109	101,410	22,178,337		**PF00534**	149,489	24,374,699
PF00905	102,112	29,424,276		**PF00441**	150,307	22,687,171
PF00582	103,451	14,218,061		**PF00990**	150,766	23,128,632
PF00300	104,359	16,171,897		**PF12796**	153,445	14,522,139
PF02775	107,283	15,693,134		**PF00196**	164,479	9,256,339
PF01047	107,718	6,321,363		**PF03989**	164,807	7,773,328
PF08241	107,758	10,138,914		**PF01381**	167,775	9,003,662
PF00563	108,082	24,522,343		**PF00077**	168,023	15,697,587
PF00593	108,668	28,476,247		**PF00550**	181,816	12,094,854
PF00202	110,266	42,269,614		**PF00575**	190,207	14,166,906
PF02321	111,802	20,897,073		**PF00171**	194,683	81,211,833
PF00067	114,258	35,047,017		**PF00270**	200,180	33,127,148
PF00248	120,944	31,729,009		**PF00583**	204,075	16,632,252
PF00378	123,859	28,325,852		**PF13193**	209,971	16,068,392
PF13414	123,888	8,403,414		**PF00392**	231,015	14,526,574
PF01370	127,456	27,264,279		**PF00069**	232,632	54,927,145
PF03144	129,814	9,044,608		**PF00486**	239,488	18,215,882
PF07715	132,540	14,480,124		**PF00501**	284,906	110,619,523
PF13855	135,318	7,952,879		**PF00571**	286,447	16,639,225
PF02770	136,802	13,413,041		**PF00440**	328,784	15,284,061
PF01380	137,800	17,567,327		**PF00072**	547,587	61,076,537
PF00291	137,827	40,668,471		**PF00005**	1,466,247	216,811,560
PF02771	145,639	16,227,629				

Chapter 3

Multiple Sequence Alignment Computation Using the T-Coffee Regressive Algorithm Implementation

---

Edgar Garriga, Paolo Di Tommaso, Cedrik Magis, Ionas Erb, Leila Mansouri, Athanasios Baltzis, Evan Floden, Cedric Notredame

Multiple Sequence Alignment. Methods in Molecular Biology, vol 2231. Humana, New York, NY. https://doi.org/10.1007/978-1-0716-1036-7_6

Chapter 6

Multiple Sequence Alignment Computation Using the T-Coffee Regressive Algorithm Implementation

Edgar Garriga, Paolo Di Tommaso, Cedrik Magis, Ionas Erb, Leila Mansouri, Athanasios Baltzis, Evan Floden, and Cedric Notredame

Abstract

Many fields of biology rely on the inference of accurate multiple sequence alignments (MSA) of biological sequences. Unfortunately, the problem of assembling an MSA is NP-complete thus limiting computation to approximate solutions using heuristics solutions. The progressive algorithm is one of the most popular frameworks for the computation of MSAs. It involves pre-clustering the sequences and aligning them starting with the most similar ones. The scalability of this framework is limited, especially with respect to accuracy. We present here an alternative approach named regressive algorithm. In this framework, sequences are first clustered and then aligned starting with the most distantly related ones. This approach has been shown to greatly improve accuracy during scale-up, especially on datasets featuring 10,000 sequences or more. Another benefit is the possibility to integrate third-party clustering methods and third-party MSA aligners. The regressive algorithm has been tested on up to 1.5 million sequences, its implementation is available in the T-Coffee package.

Key words Sequence alignment, MSA, Guide tree, Progressive alignment

1 Introduction

Multiple sequence alignment (MSA) is an NP-complete problem whose computation relies on approximate heuristic solutions. The most common solution is the progressive method [1]. This method starts by aligning the most similar sequences following a pre-computed guide tree, but the accuracy drops when dealing with a large number of sequences.

The regressive method [2] works the other way around and starts by aligning the most diverse sequences going from the root of the guide tree to the leaves. MSAs are constructed through a divide and conquer process during which smaller MSAs – named sub-MSAs – encompassing the more diverse sequences are gradually expanded until all sequences have been incorporated within the final model. Extensive benchmark analyses carried out on Homfam

Kazutaka Katoh (ed.), *Multiple Sequence Alignment: Methods and Protocols*, Methods in Molecular Biology, vol. 2231, https://doi.org/10.1007/978-1-0716-1036-7_6,

[3] and Pfam [4] have shown that the regressive algorithm is both more scalable than regular methods – it was shown to align 1.5 million sequences – and also more accurate, especially when dealing with datasets larger than 10,000 sequences.

An important characteristic of the T-Coffee implementation of this algorithm is its modularity. It allows several third-party methods to be used in order to both estimate the guide tree and to apply the most commonly used alignment algorithms – including the consistency-based version of T-Coffee [5] – to perform the sub-MSAs during the divide and conquer stage.

2 Materials

2.1 Equipment Setup

- Computer: Any computer running Linux or Mac OSX with access to the internet.
- Software: T-Coffee can be downloaded from http://tcoffee.org/Packages/Stable/Latest. It is distributed as a set of precompiled binaries for Linux and Mac OSX platforms (32-bit or 64-bit) with a guided install procedure. This is the smoothest and quickest way to install T-Coffee on a local machine, as it comes with all the required components and does not require any special user privileges. It is also possible to download the source code from GitHub or use it from Conda or Docker containers.
- Sequence to align: www.tcoffee.org/Projects/regressive/datasets/protocols.tar.gz.

2.2 Procedure

T-Coffee: obtaining and installing t-coffee

Install T-Coffee by following one of the following options, some of them are possible to run on both Linux and MacOSX operating systems (OS), and others are specific to each OS.

2.2.1 Binary

Linux

1. Download the installer package from http://tcoffee.org/Packages/Stable/Latest/linux/
2. Grant execution permission to the downloaded file with the following command:
   chmod +x T-COFFEE_installer_<version_xxx>.bin
3. Launch the installation wizard with
   ./T-COFFEE_installer_<version_xxx>.bin
4. Follow the wizard instructions and complete the installation.
5. Open a new terminal session to be sure that your environment is updated.

6. Type the following command to verify that the installation was successful:
   t_coffee -version

MacOSX

1. Download the T-COFFEE_distribution_Version <version>.tar.gz package from http://tcoffee.org/Packages/Stable/Latest/.
2. tar -cvf T-COFFEE_distribution_Version_<version>.tar.gz.
3. cd T-COFFEE_distribution_Version_<version>_.
4. type ./install all.
5. Follow the instructions of the installer to update your environement.
6. Type the following command to verify that the installation is successful:
   t_coffee –version.

2.2.2 Compilation from Source

1. Follow the instructions from the T-Coffee GitHub page: www.github.com/cbcrg/tcoffee
2. Go inside the source folder
   cd t_coffee/src
3. Compile the package with
   make t_coffee
4. Add the compile folder in your path.
   mv t_coffee /bin/

2.2.3 Docker

From the command line, you can download the docker container with the following command:

docker pull cbcrg/tcoffee_protocols

You can use the container with any of the workflow managers, or run it in an iterative mode using the command:

docker run -ti --mount type=bind,source=/<path_to_data>/, target=/<container_data_folder>/cbcrg/tcoffee_protocols

2.2.4 Conda

To install the conda package, you should download from bioconda channel with the following command:

conda create --name tcoffee_protocols -c bioconda t-coffee
conda activate tcoffee_protocols.

This package includes all the third-party software needed for this protocol, but we can always generate an environment combining T-Coffee with other software.

3 Methods

The T-Coffee regressive algorithm has been developed to allow the computation of ultra-large MSAs. The algorithm's main steps are as follows:

- 1: Computation of a rooted guide tree using any relevant method, including mBed [6] and PartTree [7] (*see* **Note 1**).
- 2: Label each node with the label of the longest sequence among its progeny (*see* Fig. 1), (i.e., the root will be labeled with the longest sequence), Fig. 1.
- 3: Starting from the root node (parent node), and going one generation at a time, collect N nodes—N is a free parameter. In Fig. 2, N is set to 3 but in practice, N is set to 1000. Its value can be changed via the parameter -**reg_nseq**

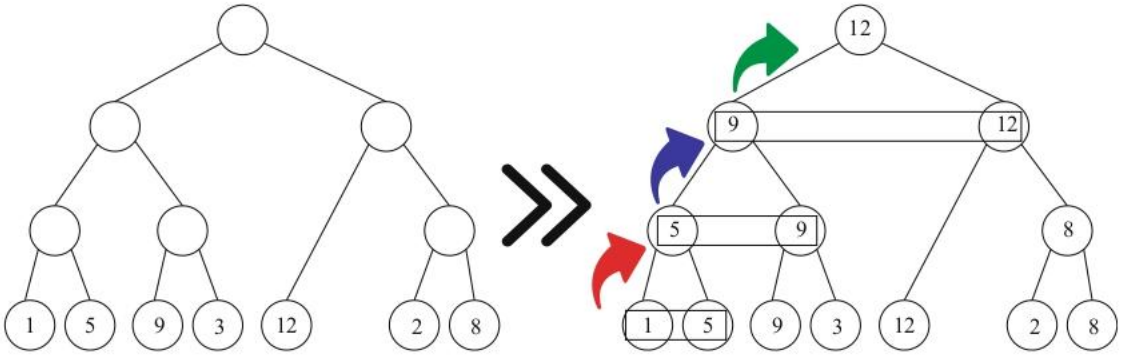


Fig. 1 From the naked guide tree, the algorithm starts from the leaf until the root labeling the internal nodes with the longest sequence of the children

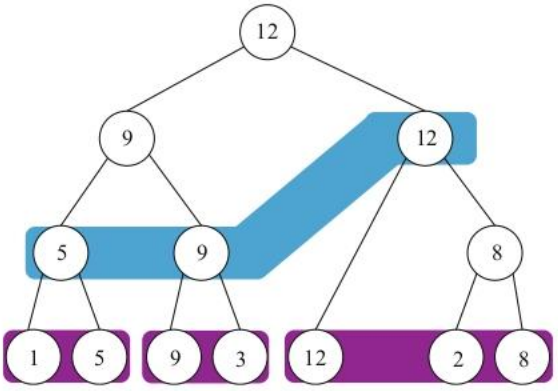


Fig. 2 From the root of the guide tree, N sequences are collected, ($N = 3$ in this example.) This is repeated recursively on the remaining nodes until all the leaf nodes are included in one of the subMSAs

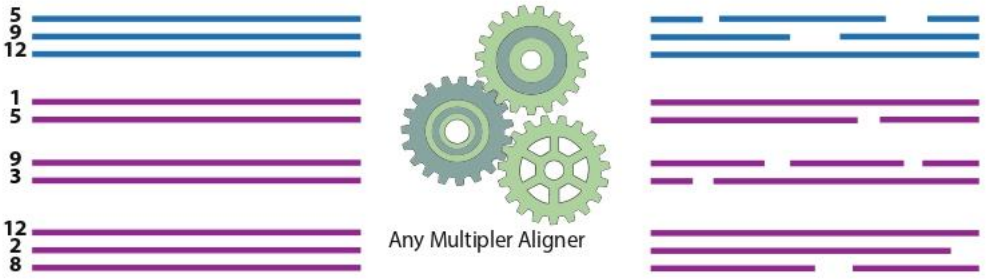


Fig. 3 Once all the small groups are defined, it is possible to generate the subMSAs in a parallel way using any third-party alignment software

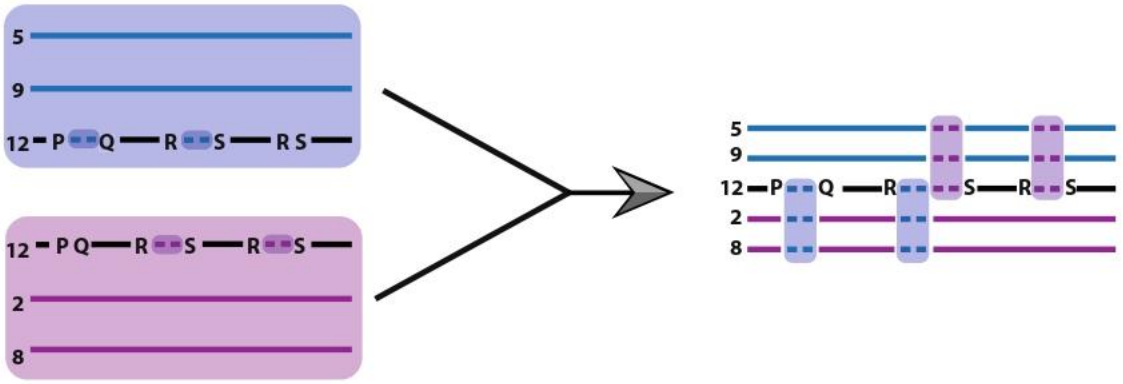


Fig. 4 Parent and children subMSAs are merged using the common sequence (12) projecting indels from parent to child and from child to parent

- 4: Carry out an MSA of the N Sequences that label the N nodes. For instance, in Fig. 2, with $N = 3$ this will involve sequences 5, 9, and 12 (blue envelope). This MSA is named a parent subMSA, and it can be computed using any third-party aligner.
- 5: Run **steps 3–4** on every node selected in #3 that is not a leaf, the resulting subMSAs will be the children MSAs of the parent subMSA computed one step earlier. For instance, in Fig. 3, the children subMSAs will be made of sequences (1,5), (9,3), and (12,2,8). The procedure stops once every leaf node has been incorporated in an MSA.
- 6: Since every MSA shares the sequence of its parent node with its parent MSA, the children and their parent subMSAs can be combined through these common sequences without the need of an extra alignment step, as shown in Fig. 4. Combining the subMSAs merely involves stacking the columns linked by their common sequence (*see* **Notes 2** and **3**).

The regressive algorithm has been shown to have exceptional scalability [2]. One of the reasons for this is the reliance on a strict divide and conquer procedure that never involves aligning more than N sequences. As a consequence, for M input sequences, the

deployment of any third-party method – regardless of its original complexity – becomes linear in time and memory as it merely involves carrying out M/N individual MSAs. Moreover, the independence of these MSAs makes their computation an embarrassingly parallel problem.

Aside from its algorithmic properties, the regressive implementation of the T-Coffee algorithm also brings many added benefits through the seamless integration of a large number of third-party clustering and alignment methods. Overall, five clustering methods are supported along with five multiple sequence aligners. The package comes along with an extensive documentation allowing non-supported alignment methods to be incorporated via simple configuration files.

In the next section we explore various combinations of clustering methods and alignment algorithms that allow users to explore different trade-offs between accuracy and efficiency. For instance, it is possible to very rapidly estimate ultra-large models by combining the fastest clustering method (PartTree) with the fastest MSA method (MAFFT default). The same framework makes it possible to combine a slower but more accurate tree method (like mBed) with a very accurate MSA method (like MAFFT-ginsi) that only allows aligning a few hundred sequences but can be massively scaled-up by the regressive framework.

3.1 Validated Method Combinations

The following combinations of pre-clustering and alignment methods were validated in the original paper for their relative speed and accuracy. They are recommended for large scale analysis.

3.1.1 Fast and Accurate

This mode offers the best trade-off between speed and accuracy. It relies on the Clustal Omega (ClustalO; Chapter 1) [3] mBed trees that were found to yield the highest accuracy on large datasets, while the combination of these guide trees with the ClustalO aligner results in alignments of reasonable accuracy.

t_coffee -reg -seq gluts.fasta -reg_nseq 1000 -reg_tree mbed -reg_method clustalo_msa -outfile gluts.aln -outtree gluts.mbed

3.1.2 Slower and More Accurate

As discussed earlier, the regressive algorithm framework can be used to deploy methods that would be prohibitive on any dataset larger than 1000 sequences. In the example below, we show how the MAFFT-ginsi method can be deployed on large datasets. On the HomFam dataset, this protocol required about 4.7 times more CPU time (as compared with the fast approximate mode using fftns1), but resulted in a 21% improvement in the number of correctly aligned columns.

t_coffee -reg -seq gluts.fasta -reg_nseq 1000 -reg_tree mbed -reg_method mafftginsi_msa -outfile gluts.aln -outtree gluts.mbed

3.1.3 Very Fast and Approximate

On the other end of the spectrum, the combination of the fastest aligner with the fastest clustering method provides the most efficient alignment method currently available. This combination is about 3 times faster than the fast and accurate ClustalO combination.

t_coffee -reg -seq gluts.fasta -reg_nseq 1000 -reg_tree parttree -reg_method mafftfftnsi_msa -outfile gluts.aln -outtree gluts.parttree

3.1.4 Further Method Combinations

A major strength of the regressive algorithm is its capacity to support any method combination of interest to the user. All these combos have not been validated so far, but they are nonetheless supported and available for exploration.

One of the main limitations of both the progressive and the regressive procedures is the generation of the guide tree because not all the clustering methods are able to handle a large number of sequences.

The Regressive method has the advantage that it allows to use any clustering method from which a tree can be obtained, making it possible to use algorithms that work well with big data.

T-Coffee offers some built-in options for building trees from a range of clustering algorithms, and they can be used with the **-reg_tree** flag.

- mbed: use mBed mode of ClustalO – Default
- cwdnd: use the quicktree mode of ClustalW
- parttree: parttree method of MAFFT—fastest option. Does not support sequences less than 6 AA long
- dpparttree: MAFFT fast clustering method
- fastparttree: MAFFT fast clustering method
- mafftdnd: default MAFFT tree—slower than the parttree modes
- fftns1dnd: Tree produced after the first iteration MAFFT fftns mode
- fftns2dnd: Tree produced after the second iteration MAFFT fftns mode
- upgma: upgma tree—warning cubic time computation
- nj: Neighbour Joining tree
- #<command>: Runs command <seq> > <tree>.
- filename: Any file in newick format. The seq file and the tree file must match

Thanks to the possibility to freely combine guide trees and alignment methods, the regressive algorithm allows the usage of highly accurate methods (limited to a small set of sequences) or less

accurate but faster methods. Users can create and explore their own combinations via the flag -**reg_method** that makes T-Coffee use a set of built-in aligners.

ktup_msa
blastp_msa
clustalo_msa
clustaloNF_msa
clustalw2_msa
clustalw_msa
uppNF_msa
upp_msa
msa_msa
dca_msa
mafftsparsecore_msa
mafftfast_msa
mafft_msa
…

The -**reg_nseq** flag is the only free parameter. This parameter defines the maximum number of sequences in the subMSAs. It allows to use more accurate methods that can only handle a limited number of sequences. There is also a tradeoff between the size of the subMSAs and the CPU time. Based on results in [3], we have defined this size to 1.000 sequences as a default value.

The optimal value may change somewhat depending on the guide tree and the alignment methods as well as the type of sequences to be aligned.

4 Notes

1. One of the possible issues of this method occurs in step #1, where the guide tree computation is required. Some of the classic methods are not able to handle large amounts of sequences and they may fail at delivering a guide tree. Yet, provided a guide tree is available, most methods can be deployed using the regressive mode of T-Coffee.
2. An important contribution to scalability results from the way the final MSA is assembled. Because it results from the combination of sub-MSAs containing a common sequence, the gaps do not need to be stored in memory, and they can be kept as

counts and eventually written on disc once the computation is finished. This allows the computation of models larger than the available RAM without any disk swapping_.

3. It is worth mentioning that the regressive implementation of T-Coffee explicitly avoids aligning non-homologous indels (i.e., indels having occurred independently according to the guide tree). These indels are concatenated rather than aligned. This process has two consequences: it can result in rather large MSAs (i.e. large number of columns), and it means that given two alternative guide trees (i.e., mBed and PartTree), the one producing the MSA containing the smallest number of gaps is probably the most accurate.

Acknowledgments

We acknowledge Des Higgins and Olivier Gascuel for useful discussions and feedback.

References

1. Hogeweg P, Hesper B (1984) The alignment of sets of sequences and the construction of phyletic trees: an integrated method. J Mol Evol 20:175–186. https://doi.org/10.1007/bf02257378
2. Garriga E, Di Tommaso P, Magis C et al (2019) Large multiple sequence alignments with a root-to-leaf regressive method. Nat Biotechnol 37 (12):1466–1470
3. Sievers F, Wilm A, Dineen D et al (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal omega. Mol Syst Biol 7:539. https://doi.org/10.1038/msb.2011.75
4. Finn RD, Bateman A, Clements J et al (2014) Pfam: the protein families database. Nucleic Acids Res 42:D222–D230. https://doi.org/10.1093/nar/gkt1223
5. Notredame C, Higgins DG, Heringa J (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment. J Mol Biol 302:205–217. https://doi.org/10.1006/jmbi.2000.4042
6. Blackshields G, Sievers F, Shi W et al (2010) Sequence embedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol Biol 5:21. https://doi.org/10.1186/1748-7188-5-21
7. Katoh K, Toh H (2007) PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. Bioinformatics 23:372–374. https://doi.org/10.1093/bioinformatics/btl592

Chapter 4

Nextflow integration for the Research Object Specification

Edgar Garriga, Paolo Di Tommaso, Cedric Notredame

Zenodo https://doi.org/10.5281/zenodo.1323831

Nextflow integration for the Research Object Specification

Edgar Garriga Nogales
Centre for Genomic Regulation (CRG)
The Barcelona Institute for Science and Technology
Barcelona, Spain
edgar.garriga@crg.eu

Paolo Di Tommaso
Centre for Genomic Regulation (CRG)
The Barcelona Institute for Science and Technology
Barcelona, Spain
paolo.ditommaso@crg.eu

Cedric Notredame
Centre for Genomic Regulation (CRG)
The Barcelona Institute for Science and Technology
Barcelona, Spain
cedric.notredame@crg.eu

Keywords— nextflow, research object, reproducibility, provenance

I. Introduction

For reproducibility criteria to be met in a scientific context an increasing number of conditions need to be fulfilled. These conditions, explicited by the FAIR principle include traceability, reusability and data/methods permanent availability (findable). The challenge is not only to keep the right elements bundled together, but also to keep track of each component individual history (including individual updates) while associating every computational analysis with a transparent source. This issue, known as provenance, is the one we have been addressing in the context of this proposal. It is increasingly critical, at a time when a growing number of computational procedures are used to assess medical risks and take therapeutic decision. Our solution involves using the Research Object (RO)[1] specification that have allowed us to implement a method that enables the creation of a package collecting all provenance metadata of a computational experiment, so that it can be easily shared, archived and reproduced when needed.

In practice, keeping all the required information bundled together (paper, slides, methods, pipelines, etc) can be challenging, especially when adding the constraint of fine grain querying. The aim of this this proposal is to create a package based on the Research Object specification. Thanks to this procedure, all the provenance information of a computation experiment that can be easily collected, shared and archived

We show here how a slight adaptation of a workflow tool like Nextflow[2] can allow for the seamless transfer of unique ID tags to various elements of data thus making it easier to trace the data and its associated objects for citation purposes.

II. Method

Nextflow is a framework based on the dataflow programming model, which simplifies the writing of parallel and distributed pipelines. Given a multi-step pipeline, Nextflow allows explicit dependencies to be declared between tasks thus allowing output and input to be piped across the workflow, with specific operation possibly carried out between tasks (merge, sort, split, etc…). The tasks themselves are usually encapsulated in containers and deployed by Nextflow across computational platforms (Amazon cloud, legacy batch schedulers,Kubernetes, etc.). Being able to decompose a pipeline into multiple processes, possibly written in different scripting languages (Bash, Perl, Python, etc.) simplifies the pipeline development. A major advantage of Nextflow is its ability to deploy the execution of a pipeline across multiple platforms.

Research Object is a method for the identification, aggregation, and exchange of information. The primary goal is to provide a way of associating together related resources from the same project, (i.e. the pipeline, auxiliar scripts, data, slides or the final article).

The Research Object concept is motivated by a desire to improve the reproducibility of computational methods and experiments. Its main three principles are: *Identity*, providing a unique identifier to the project, as the DOI for the publication or the ORCID for the scientist. *Aggregation*, allowing the author to wrap all the elements used for the project (i.e. slides, article, scripts, etc.). With the Research Object, we can share all the elements of a project together with the same ID. Finally, the last main principle of

Research Object is the *Annotation*, a specific layer of metadata that explicitly defines the relation between elements, as well as their time and mode of production[3]. As such, the RO technology allows having in the same package human and computer readable data while making the projects traceable and FAIR compliant.

III. Implementation

The integration will make Nextflow able to produce a zip file with the following structure:

-A *Data* folder, with all the input files of our pipeline.

-The *Workflow* folder containing all the files of the pipeline base directory (e.g. the main Nextflow script or the config file of the pipeline).

-The *Snapshot* folder is the one used re-execute the pipeline with the same parameters if its needed.

-The *Metadata* folder contains the log file of the past executions, the metadata file with information about the container used and the Nextflow version. The metadata folder contains the provenance file too. With this file we can see how and when the intermediate files where generated, and which process used them as an input.

-The *Output* folder, where the output of the execution is stored.

-The *RO* folder, containing RO's manifest with information about the author and the creation of the RO.

IV. Results

The Nextflow-RO integration allows the creation of an RO package when executing a Nextflow workflow. The result is a single package with a unique identification (identity) containing all the important files of the project (aggregation) like the metadata, logs, results and the workflow directory. Another important value of this integration is the generation of the provenance annotation. This information makes it easier to share and reproduce an analysis. It also increases the transparency on the procedure behind the analysis. These three elements contribute towards the three core principles of RO.

The main current limitation involve capturing all the relevant data in a non-user dependent way. Another issue is the burden of metadata capture on the user side, since even small scale analysis can easily generate metadata with a size larger than both the raw data and the final analysis. In a future work we plan to continue developing this feature, evolving as much as possible with the community's feedback, we will try to increase the provenance level and make the generation process as user-friendly as possible.

References

[1] Khalid Belhajjame, Jun Zhao, Daniel Garijo, Matthew Gamble, Kristina Hettne, Raul Palma, Eleni Mina, Oscar Corcho, José Manuel Gómez-Pérez, Sean Bechhofer, Graham Klyne, Carole Goble (2015) Using a suite of ontologies for preserving workflow-centric research objects, Web Semantics: Science, Services and Agents on the World Wide Web, https://doi.org/10.1016/j.websem.2015.01.003

[2] P. Di Tommaso, et al. Nextflow enables reproducible computational workflows. Nature Biotechnology 35, 316–319 (2017)

[3] Farah Zaib Khan, Stian Soiland-Reyes, Michael R. Crusoe, Andrew Lonie, & Richard O. Sinnott. (2018). CWLProv - Interoperable Retrospective Provenance capture and its challenges. Zenodo. http://doi.org/10.5281/zenodo.1215611

Chapter 5: Discussion

- **Large multiple sequence alignments with a root-to-leaf regressive method**

Multiple sequence alignments are required for various biological tasks such as functional predictions, structural modeling, and phylogenetic inference. As the size of the datasets utilized in these applications grows, so must the methodologies used to analyze them. A novel agglomerative multiple sequence alignment technique is described whose scaling up capacities beat all known methods in terms of accuracy in **chapter 2**, regressive computing of large scale multiple sequence alignments.

The challenge of calculating proper multiple sequence alignments is NP-complete. Because there is no guarantee of a precise result, all known solutions are based on approximate heuristics. The use of heuristics necessitates revisiting and readapting these procedures whenever the nature of the problem changes, even if only a little. For example, the growing demand for progressively massive datasets has revealed an unanticipated restriction of the current alignment architecture, known as progressive alignment, in recent years. When the number of sequences exceeds a thousand homologs, contrary to popular belief, alignment accuracy declines.

This discovery came as a complete surprise because it had long been known that its relative alignment accuracy would improve when a set of sequences was embedded within a bigger dataset. This constraint is significant since it pulls the existing MSA scaling paradigm to a dead-end road. Furthermore, it raises substantial issues about our ability to properly integrate the biological data generated by the new genome projects.

I provide a straightforward and incredibly successful way to scale-up MSA modeling methods referred to as regressive by referring to the progressive algorithm. Sequences are clustered using a guide tree that dictates the order in which they will be aligned when executing a progressive (or regressive) alignment. The progressive alignments begin with the most similar sequences - sister leaves - and work their way down to the root. The regressive solution uses the same guide-tree, but instead of going from leaf to root, we employ it to collect the most diverse sequences and then start aligning them; the same algorithm is then reapplied as we get closer to the root. All subsequent alignments are grafted onto the scaffold of the initial alignment, which contains the most varied sequences. The implemented technique enables the usage of typical large-scale aligners such as Mafft (Katoh et al., 2002), ClustalO (Sievers et al., 2011), and UPP (Nguyen et al., 2015) in both a

regressive and progressive manner for validation purposes. This strategy allows us to separate each algorithmic component's contribution and conclude that the regressive solution is preferable to the progressive option. On the 20 biggest reference datasets in HOMFAM, the regressive method outperforms the progressive approach by around 6.5 percentage points on average (10,000 to 93,000 sequences). More crucially, the regressive framework's improved scalability allowed us to use small-scale, very accurate approaches like mafft-ginsi on huge datasets for which they were not designed. The alignments produced are the most precise ever reported on these datasets. This result is of immediate practical utility to the community because it comes with a mature production software created in T-Coffee (Notredame et al., 2000), and it is available on GitHub.

On the other hand, the regressive algorithm is much more than a new piece of software.
By clearly separating the guide-tree and the aligners, the regressive method redefines the scope of research in multiple sequence alignment computing. It permits the implementation of an apparent contradiction between the development of increasingly faster and more exact clustering algorithms on the one hand, and the construction of very accurate small scale aligners on the other. The two independent groups will contribute their distinct strengths and develop fresh approaches whose availability is crucial for the future of biology, it is claimed, by explicitly severing the relationship between alignment and clustering.

- **Multiple Sequence Alignment Computation Using the T-Coffee Regressive Algorithm Implementation**

Complex pipelines containing multiple third-party tools, with many dependencies on external scripts, libraries, environmental variables, etc., must be built to undertake large-scale comparative bioinformatics analyses. Due to the volume of data they must process, these pipelines demand high-performance computing (HPC) resources, which means they must frequently be transported from a laptop to a powerful desktop computer, a cluster, or even a cloud or supercomputer. Routine bioinformatics analysis is difficult to reproduce due to these reasons plus a mix of difficult-to-control factors.
Developing a new paradigm to perform MSAs where the target is the alignment of millions of sequences is great, but it's better to allow people to use it. And this is the goal of this project of **chapter 3**. The idea is to build solutions for easy usage of the Regressive method

and add the extra value of reproducibility, portable, and with the capacity to scale with zero effort thanks to the workflow manager Nextflow (Di Tommaso et al., 2017).

One of the learnings of this project is the necessity of good maintenance of the software. During the process of building the containers and standardization of the methods, some problems appeared. Software without maintenance, incompatibility of some programs with today's technologies, or software without documentation or behind paywalls have become unusable. For example, the Regressive methodology opens the door to use aligners very accurately but only with the ability to align a few sentences. These aligners were developed decades ago, and they were a bit "forgotten in a drawer". Now it's time to explode their accuracy. Still, it has been impossible or very hard during this project. These difficulties go beyond the idea of the need for this effort of containers and good documentation or a manuscript as the one published.

- **Nextflow integration for the Research Object Specification**

The concept of reproducible genomic analysis becomes more important than ever at a time when the precision medicine program is about to introduce the systematic use of -omics data in our daily lives. Unfortunately, wet lab experiment variation is frequently thought to be the cause of reproducibility concerns. Nextflow is a mechanism for managing computational workflows that offers a simple and efficient solution to this issue. It is demonstrated how Nextflow allows current pipelines to be deployed efficiently and stably, providing a long-awaited solution to the problem of ensuring computational reproducibility when doing -omics data processing.

Thanks to the Google Summer of Code, it was possible to add some functionality to Nextflow. The project is explained in **chapter 4**, and the idea is to produce a Research Object (RO) (Belhajjame et al., 2015). Once finished the project, the most exciting part was not only the RO by itself. The idea of provenance in Nextflow is quite interesting. Adding this extra value thanks to the metainformation is desirable. It is being able to know 'who' ran the analysis, in which environment, when, and the exact version of the software and the parameters. This adds a level to the reproducibility schema and can be helpful in personalized medicine.

One of the needs discovered in this project was the GUI for provenance. RO was not meant to show the data and file in a friendly way. But provenance needs a bit of work because Nextflow can produce a massive amount of metainformation (for each process,

each task, workflow run, etc.) a. Itld be very useful/powerful with an appropriate interface. It could be a web interface thanks to the functionality of Tower (tower.nf) or the great community of nf-core (Ewels et al., 2020)

Bibliography

Apostolico, A., & Galil, Z. (Eds.). (1997). *Pattern Matching Algorithms*. Oxford University Press. https://doi.org/10.1093/oso/9780195113679.001.0001

Baeza-Yates, R., & Navarro, G. (1996). A faster algorithm for approximate string matching. In D. Hirschberg & G. Myers (Eds.), *Combinatorial Pattern Matching* (pp. 1–23). Springer. https://doi.org/10.1007/3-540-61258-0_1

Baeza-Yates, R., & Navarro, G. (1998). Fast approximate string matching in a dictionary. *Proceedings. String Processing and Information Retrieval: A South American Symposium (Cat. No.98EX207)*, 14–22. https://doi.org/10.1109/SPIRE.1998.712978

Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., Mina, E., Corcho, O., Gómez-Pérez, J. M., Bechhofer, S., Klyne, G., & Goble, C. (2015). Using a suite of ontologies for preserving workflow-centric research objects. *Journal of Web Semantics*, *32*, 16–42. https://doi.org/10.1016/j.websem.2015.01.003

Blackshields, G., Sievers, F., Shi, W., Wilm, A., & Higgins, D. G. (2010). Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, *5*(1), 21. https://doi.org/10.1186/1748-7188-5-21

Boytsov, L. (2011). Indexing methods for approximate dictionary searching: Comparative analysis. *ACM Journal of Experimental Algorithmics*, *16*, 1.1:1.1-1.1:1.91. https://doi.org/10.1145/1963190.1963191

Burrows, M., & Wheeler, D. (1994). A block-sorting lossless data compression algorithm. *Digital SRC Research Report*.

Chang, J.-M., Di Tommaso, P., & Notredame, C. (2014). TCS: A new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. *Molecular Biology and Evolution*, *31*(6), 1625–1637. https://doi.org/10.1093/molbev/msu117

Chatzou, M., Magis, C., Chang, J.-M., Kemena, C., Bussotti, G., Erb, I., & Notredame, C. (2016). Multiple sequence alignment modeling: Methods and applications. *Briefings in Bioinformatics*, *17*(6), 1009–1023. https://doi.org/10.1093/bib/bbv099

Davidson, S., Cohen-Boulakia, S., Eyal, A., Ludascher, B., McPhillips, T., Bowers, S., Anand, M. K., & Freire, J. (2007). *Provenance in Scientific Workflow Systems*. 7.

Dayhoff, M. O., & Schwartz, R. M. (1978). A model of evolutionary change in proteins. *In Atlas of Protein Sequence and Structure*.

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, *35*(4), 316–319. https://doi.org/10.1038/nbt.3820

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*. https://doi.org/10.1007/BF01386390

Do, C. B., Mahabhashyam, M. S. P., Brudno, M., & Batzoglou, S. (2005). ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, *15*(2), 330–340. https://doi.org/10.1101/gr.2821705

Edgar, R. C., & Batzoglou, S. (2006). Multiple sequence alignment. *Current Opinion in Structural Biology*, *16*(3), 368–373. https://doi.org/10.1016/j.sbi.2006.04.004

Ewels, P. A., Peltzer, A., Fillinger, S., Patel, H., Alneberg, J., Wilm, A., Garcia, M. U., Di Tommaso, P., & Nahnsen, S. (2020). The nf-core framework for community-curated bioinformatics pipelines. *Nature Biotechnology*, *38*(3), 276–278. https://doi.org/10.1038/s41587-020-0439-x

Garriga, E., Di Tommaso, P., Magis, C., Erb, I., Mansouri, L., Baltzis, A., Laayouni, H., Kondrashov, F., Floden, E., & Notredame, C. (2019). Large multiple sequence alignments with a root-to-leaf regressive method. *Nature Biotechnology*, *37*(12), 1466–1470. https://doi.org/10.1038/s41587-019-0333-6

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, *162*(3), 705–708. https://doi.org/10.1016/0022-2836(82)90398-9

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press. https://doi.org/10.1017/CBO9780511574931

Henikoff, S., & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, *89*(22), 10915–10919. https://doi.org/10.1073/pnas.89.22.10915

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, *18*(6), 341–343. https://doi.org/10.1145/360825.360861

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Katoh, K., Misawa, K., Kuma, K., & Miyata, T. (2002). MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, *30*(14), 3059–3066. https://doi.org/10.1093/nar/gkf436

Katoh, K., & Toh, H. (2007). PartTree: An algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics (Oxford, England)*, *23*(3), 372–374. https://doi.org/10.1093/bioinformatics/btl592

Kececioglu, J. (1993). The maximum weight trace problem in multiple sequence alignment. *Combinatorial Pattern Matching - 4th Annual Symposium, CPM 1993, Proceedings*, 106–119. https://doi.org/10.1007/bfb0029800

Kececioglu, J. D., Lenhof, H. P., Mehlhorn, K., Mutzel, P., Reinert, K., & Vingron, M. (2000). A polyhedral approach to sequence alignment problems. *Discrete Applied Mathematics*, *104*(1–3), 143–186. https://doi.org/10.1016/S0166-218X(00)00194-3

Kemena, C., & Notredame, C. (2009). Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, *25*(19), 2455–2465. https://doi.org/10.1093/bioinformatics/btp452

Kurtzer, G. M., Sochat, V., & Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PloS One*, *12*(5), e0177459. https://doi.org/10.1371/journal.pone.0177459

Liu, Y., Schmidt, B., & Maskell, D. L. (2010). MSAProbs: Multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. *Bioinformatics (Oxford, England)*, *26*(16), 1958–1964. https://doi.org/10.1093/bioinformatics/btq338

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2:2.

Michener, C. D., & Sokal, R. R. (1957). A Quantitative Approach to a Problem in Classification. *Evolution*, *11*(2), 130–162. https://doi.org/10.1111/j.1558-5646.1957.tb02884.x

Myers, E. W., & Miller, W. (1988). Optimal alignments in linear space. *Computer Applications in the Biosciences: CABIOS*, *4*(1), 11–17. https://doi.org/10.1093/bioinformatics/4.1.11

Myers, G. (1999). A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM*, *46*(3), 395–415. https://doi.org/10.1145/316542.316550

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, *48*(3), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

Nguyen, N. D., Mirarab, S., Kumar, K., & Warnow, T. (2015). Ultra-large alignments using phylogeny-aware profiles. *Genome Biology*, *16*(1), 124. https://doi.org/10.1186/s13059-015-0688-z

Notredame, C., Higgins, D. G., & Heringa, J. (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, *302*(1), 205–217. https://doi.org/10.1006/jmbi.2000.4042

O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D. G., & Notredame, C. (2004). 3DCoffee: Combining protein sequences and structures within multiple sequence alignments. *Journal of Molecular Biology*, *340*(2), 385–395. https://doi.org/10.1016/j.jmb.2004.04.058

Rost, B. (1999). Twilight zone of protein sequence alignments. *Protein Engineering*, *12*(2), 85–94. https://doi.org/10.1093/protein/12.2.85

Saitou, N., & Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, *4*(4), 406–425. https://doi.org/10.1093/oxfordjournals.molbev.a040454

Sankoff, D. (1975). Minimal Mutation Trees of Sequences. *SIAM Journal on Applied Mathematics*, *28*(1), 35–42.

Sellers, P. H. (1974). On the Theory and Computation of Evolutionary Distances. *SIAM Journal on Applied Mathematics*, *26*(4), 787–793. https://doi.org/10.1137/0126070

Sievers, F., Dineen, D., Wilm, A., & Higgins, D. G. (2013). Making automated multiple alignments of very large numbers of protein sequences. *Bioinformatics*, *29*(8), 989–995. https://doi.org/10.1093/bioinformatics/btt093

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., Thompson, J. D., & Higgins, D. G. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, *7*, 539. https://doi.org/10.1038/msb.2011.75

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, *147*(1), 195–197. https://doi.org/10.1016/0022-2836(81)90087-5

Thompson, J. D., Higgins, D. G., & Gibson, T. J. (1994). CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, *22*(22), 4673–4680.

Thompson, J. D., Linard, B., Lecompte, O., & Poch, O. (2011). A Comprehensive Benchmark Study of Multiple Sequence Alignment Methods: Current Challenges and Future Perspectives. *PLOS ONE*, *6*(3), e18093. https://doi.org/10.1371/journal.pone.0018093

Van Noorden, R., Maher, B., & Nuzzo, R. (2014). The top 100 papers. *Nature*, *514*(7524), 550–553. https://doi.org/10.1038/514550a

Wallace, I. M., O'Sullivan, O., Higgins, D. G., & Notredame, C. (2006). M-Coffee: Combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Research*, *34*(6), 1692–1699. https://doi.org/10.1093/nar/gkl091

Wang, L., & Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, *1*(4), 337–348. https://doi.org/10.1089/cmb.1994.1.337

Wilm, A., Higgins, D. G., & Notredame, C. (2008). R-Coffee: A method for multiple alignment of non-coding RNA. *Nucleic Acids Research*, *36*(9), e52. https://doi.org/10.1093/nar/gkn174

Yamada, K. D., Tomii, K., & Katoh, K. (2016). Application of the MAFFT sequence alignment program to large data-reexamination of the usefulness of chained guide trees. *Bioinformatics (Oxford, England)*, *32*(21), 3246–3251. https://doi.org/10.1093/bioinformatics/btw412