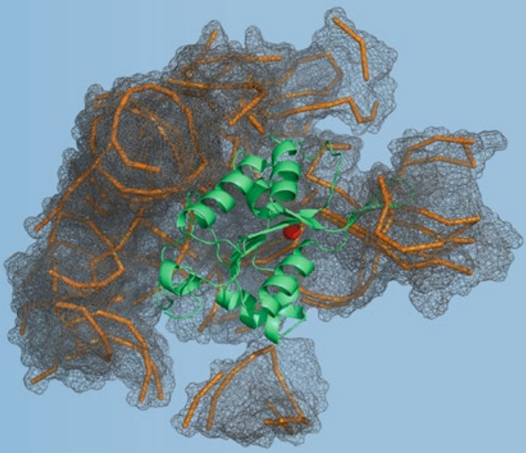
Methods in Molecular Biology 2231

Springer Protocols



Kazutaka Katoh *Editor*

Multiple Sequence Alignment

Methods and Protocols

Humana Press

Methods in Molecular Biology

Series Editor
John M. Walker
School of Life and Medical Sciences
University of Hertfordshire
Hatfield, Hertfordshire, UK

Multiple Sequence Alignment

Methods and Protocols

Edited by

Kazutaka Katoh

Research Institute for Microbial Disease, Osaka University, Osaka, Japan

Humana Press

Editor
Kazutaka Katoh
Research Institute for Microbial Disease
Osaka University
Osaka, Japan

ISSN 1064-3745   ISSN 1940-6029 (electronic)
Methods in Molecular Biology
ISBN 978-1-0716-1035-0   ISBN 978-1-0716-1036-7 (eBook)
https://doi.org/10.1007/978-1-0716-1036-7

This Humana imprint is published by the registered company Springer Science+Business Media, LLC, part of Springer Nature.
The registered company address is: 1 New York Plaza, New York, NY 10004, U.S.A.

Preface

Introduction

This volume is a collection of protocols to install and run tools for calculation and visualization of multiple sequence alignment (MSA) and other analyses related to MSA. Following the policy of *Methods in Molecular Biology*, each chapter basically consists of a brief background and a step-by-step guide for installation and actual analyses. Practical advice is also given in the Notes section in most chapters. The main target audience is experimental biologists who want to run MSA tools themselves. By reading a chapter in this volume, a reader should easily acquire basic usage of the associated tool. Detailed background and advanced usage are described in papers and/or webpages mentioned in each chapter.

Basic Concepts and Terms

To provide background for these chapters, this section explains some technical terms and established techniques that are commonly used.

Sequence alignment is an estimation of corresponding sites in a set of biological sequences. The correspondence is, in many cases, based on homology, i.e., shared ancestry in evolutionary history of the sequences compared. Alignment of just two sequences is called "pairwise sequence alignment," and alignment of three or more sequences is called "multiple sequence alignment," MSA, the theme of this volume. There is a long history of the study of sequence alignment methods, starting from the first application of the dynamic programming (DP) algorithm to pairwise alignment by Needleman and Wunsch [1]. This method gives the optimum pairwise alignment under a reasonable scoring system. It is theoretically possible to extend the DP algorithm to three or more sequences, if the score is derived as the summation of pairwise alignment scores. However, this calculation requires unpractical computational resources. In addition, for a real MSA, it is also usually necessary to consider the relationships between sequences in order to reflect their evolutionary history. Thus, the simple extension of the DP algorithm for MSA is rarely used. Instead, various heuristics are used. Representative heuristic techniques are outlined below.

Progressive Method A reasonable and widely used heuristic is the progressive method [2–4]. In this strategy, a tentative tree, called a "guide tree," is built based on an all-to-all approximate comparison. Then, the sequences are aligned from the leaves to the root on the tree, in a group-to-group manner. When the calculation reaches the root, the full MSA is obtained. Many MSA programs use the progressive method as a part of the calculation. Among them, PRANK (Chapter 2) has a notable point that it rigorously considers insertions and deletions on the guide tree.

Iterative Refinement The progressive method has a well-known problem that errors can occur in early steps (i.e., close to a leaf) of the guide tree, and those errors remain in the final step (the root of the guide tree). One effective solution is to correct this type of mistake is iterative refinement [5–7]. The procedure is: (i) construct an initial MSA; (ii) divide the

MSA into two groups; (iii) re-align the two groups; repeat (ii) and (iii). This technique is used in Prrn5 (Chapter 5), Clustal Omega (Chapter 1), MAFFT (Chapter 11), and MSA-Probs (Chapter 3).

Consistency Another important idea to overcome the limitations of the progressive method is consistency [8, 9]. In the tree-based consistency transformation technique, proposed first in Notredame et al. [10], when aligning two sequences (*A* and *B*), other sequences (e.g., *C*) in the dataset are also used. That is, in addition to direct alignment between *A* and *B*, an alternative alignment between *A* and *B* is synthesized by using alignments *AC* and *BC*. Alignment *AB* is recalculated by considering such alternative alignments and used in the progressive alignment step. As a result, alignment errors in early steps are efficiently suppressed. This method was further elaborated to use probabilistic pairwise alignments by a pair hidden Markov model in ProbCons [11]. MSAprobs (Chapter 3) represents this type of MSA method and gives highly accurate MSAs.

Specific or Newly Emerging Problems

There are a lot of useful tools for computing MSAs that are not covered here. This volume introduces several characteristic tools that are quite useful in specific situations, as different types of MSA calculations are becoming necessary.

For example, when aligning genomic sequences of protein-coding regions, both amino acid sequences and DNA sequences have to be taken into account. Spalm (Chapter 5) and MACSE (Chapter 4) are useful for this type of problem.

Now we have much more data than before because of advances of sequencing technologies. The number of sequences in a typical MSA is accordingly becoming larger. To calculate such large MSAs, a number of methods have been developed, such as the regressive option of T-Coffee (Chapter 6) and PASTA/UPP (Chapter 7). Generally, it is difficult to apply high-cost methods to such data. To optimize the balance between accuracy and computational cost, these methods combine basic methods mentioned in the previous section and newly developed techniques. Alignment-free methods have also been studied as an alternative way to compare a large number of sequences. The SpaM series (Chapter 8) provides alignment-free comparisons for various purposes, including phylogeny of unassembled reads.

At this point, large-scale sequence data usually has many errors due to limitations in sequencing quality. Such errors naturally have a negative effect on downstream analyses. Lamassemble (Chapter 9) aligns and merges long raw reads of low quality to infer a more accurate sequence. Based on a given model of error patterns, it produces better MSAs and a consensus sequence for low-quality reads than general purpose MSA tools. Another solution is to detect and filter out problematic regions; PREQUAL (Chapter 10) takes this strategy, which is applicable to general datasets with unknown error patterns.

MAFFT-DASH (Chapter 11) and Mustguseal (Chapter 12) utilize protein structural information. This approach is useful in functional analyses using remote protein homologs, as protein structure tends to be more conserved over long evolutionary timescales even when homology at the sequence level is no longer detectable.

Finally, visualization and summarization of data is also becoming important since we have much more sequence data than before. Four alignment viewers/editors, Jalview (Chapter 13), Wasabi (Chapter 14), Seaview (Chapter 15), and NCBI genome workbench

(Chapter 16), are included in this volume. These chapters mainly describe functions related to MSA, but these tools have many other features for interpreting and utilizing MSAs.

Since MSA is related to a wide range of biological problems, this volume cannot cover all important issues. First, this volume mainly covers MSA of a single gene. Analysis of genomic data is only briefly mentioned in Chapters 5, 9, and 16. Second, since MSA is tightly related to estimation of evolutionary history as noted earlier, a possible direction for improvement of MSA is co-estimation of tree and alignment such as BAli-Phy [12] and StatAlign [13]. These types of methods are not included in this volume but have been very recently described in another volume in this series [14]. There is a controversy about the relationship between "evolutionarily correct" and "structurally correct" MSAs, as discussed in Chapters 2 and 17. That is, methods strictly based on an evolutionary model do not necessarily accurately align residues that are thought to be structurally conserved. There are several different interpretations of this observation, and this is an important issue for further development of MSA methods.

Osaka, Japan ***Kazutaka Katoh***

References

1. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48: 443–453.
2. 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.
3. Feng DF, Doolittle RF (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J Mol Evol 25: 351–360.
4. Higgins DG, Sharp PM (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene 73: 237–244.
5. Barton GJ, Sternberg MJ (1987) A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons. J Mol Biol 198: 327–337.
6. Berger MP, Munson PJ (1991) A novel randomized iterative strategy for aligning multiple protein sequences. Comput Appl Biosci 7: 479–484.
7. Gotoh O (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. J Mol Biol 264(4): 823–838.
8. Gotoh O (1990) Consistency of optimal sequence alignments. Bull Math Biol 52: 509–525.
9. Kececioglu J (1993) The maximum weight trace problem in multiple sequence alignment In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, (eds.), Combinatorial pattern matching. Berlin, Heidelberg: Springer Berlin Heidelberg pp. 106–119
10. 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.
11. Do CB, Mahabhashyam MS, Brudno M, Batzoglou S (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. Genome Res 15: 330–340.
12. Suchard MA, Redelings BD (2006) BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22: 2047–2048.
13. Novak A, Miklos I, Lyngso R, Hein J (2008) StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. Bioinformatics 24: 2403–2404.
14. Herman JL (2019) Enhancing statistical multiple sequence alignment and tree inference using structural information. Methods Mol Biol 1851: 183–214.

Contents

Contributors

Athanasios Baltzis • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

Daniel Barton • *University of Dundee, Dundee, Scotland, UK; Institute of Physics, Chinese Academy of Sciences, Beijing, China*

Geoffrey J. Barton • *University of Dundee, Dundee, Scotland, UK*

Colleen J. Bollin • *National Center for Biotechnology Information, U.S. National Library of Medicine, North Potomac, MD, USA*

Fabien Burki • *Department of Organismal Biology (Program in Systematic Biology), Uppsala University, Uppsala, Sweden; Science for Life Laboratory, Uppsala University, Uppsala, Sweden*

G. Mungo Carstairs • *University of Dundee, Dundee, Scotland, UK*

Nathalie Chantret • *AGAP, University of Montpellier, CIRAD, INRA, Montpellier SupAgro, Montpellier, France*

Nicolas Comte • *INRIA Grenoble-Rhône-Alpes, Montbonnot, France*

Frédéric Delsuc • *Institut des Sciences de l'Evolution de Montpellier (ISEM), CNRS, IRD, EPHE, Université de Montpellier, Montpellier, France*

Paolo Di Tommaso • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

Suzanne Duce • *University of Dundee, Dundee, Scotland, UK*

Ionas Erb • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

Evan Floden • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

Martin C. Frith • *Artificial Intelligence Research Center, AIST, Tokyo, Japan; Graduate School of Frontier Sciences, University of Tokyo, Chiba, Japan; Computational Bio Big-Data Open Innovation Laboratory, AIST, Tokyo, Japan*

Edgar Garriga • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

Jorge González-Domínguez • *Computer Architecture Group, Universidade da Coruña, CITIC, A Coruña, Spain*

Osamu Gotoh • *Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), AIST Tokyo Waterfront Bio-IT Research Center, Tokyo, Japan; Department of Computational Biology and Medical Sciences, The University of Tokyo, Chiba, Japan*

Manolo Gouy • *Université de Lyon, Université Lyon 1, CNRS, Laboratoire de Biométrie et Biologie Evolutive UMR 5558, Villeurbanne, France*

Desmond G. Higgins • *School of Medicine, University College Dublin, Dublin, Ireland*

Iker Irisarri • *Department of Applied Bioinformatics, Institute for Microbiology and Genetics, University of Göttingen, Göttingen, Germany; Department of Organismal Biology (Program in Systematic Biology), Uppsala University, Uppsala, Sweden;*

Department of Biodiversity and Evolutionary Biology, Museo Nacional de Ciencias Naturales, Madrid, Spain

KAZUTAKA KATOH • *Research Institute for Microbial Diseases, Osaka University, Osaka, Japan; Immunology Frontier Research Center, Osaka University, Osaka, Japan; Artificial Intelligence Research Center, AIST, Tokyo, Japan*

ANATOLIY KUZNETSOV • *National Center for Biotechnology Information, U.S. National Library of Medicine, North Potomac, MD, USA*

SONGLING LI • *Research Institute for Microbial Diseases, Osaka University, Osaka, Japan*

ARI LÖYTYNOJA • *Institute of Biotechnology, HiLIFE, University of Helsinki, Helsinki, Finland*

LAUREN LUI • *University of Dundee, Dundee, Scotland, UK; UC Santa Cruz, Santa Cruz, CA, USA*

CEDRIK MAGIS • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

LEILA MANSOURI • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain*

DAVID M. A. MARTIN • *University of Dundee, Dundee, Scotland, UK*

ANNE MENARD • *Labware Inc., Wilmington, DE, USA*

SIAVASH MIRARAB • *Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA, USA*

SATOMI MITSUHASHI • *Department of Human Genetics, Yokohama City University Graduate School of Medicine, Yokohama, Japan; Department of Genomic Function and Diversity, Tokyo Medical and Dental University, Tokyo, Japan*

BURKHARD MORGENSTERN • *Department of Bioinformatics (IMG), University of Göttingen, Göttingen, Germany*

KIRA MOURÃO • *University of Dundee, Dundee, Scotland, UK; Synpromics Ltd., Edinburgh, Scotland, UK*

CEDRIC NOTREDAME • *Centre for Genomic Regulation, The Barcelona Institute of Science and Technology, Barcelona, Spain; Universitat Pompeu Fabra (UPF), Barcelona, Spain*

T. CHARLES OFOEGBU • *University of Dundee, Dundee, Scotland, UK; Capgemini, Paris, France*

DAVID P. PARSONS • *INRIA Grenoble-Rhône-Alpes, Montbonnot, France*

JAMES B. PROCTER • *University of Dundee, Dundee, Scotland, UK*

VINCENT RANWEZ • *AGAP, University of Montpellier, CIRAD, INRA, Montpellier SupAgro, Montpellier, France*

DAVID ROLDAN-MARTINEZ • *Capgemini, Paris, France; Universitad Politècnica de Valencia, Valencia, Spain*

JOHN ROZEWICKI • *Research Institute for Microbial Diseases, Osaka University, Osaka, Japan; Immunology Frontier Research Center, Osaka University, Osaka, Japan*

YANA SHARAPOVA • *Belozersky Institute of Physicochemical Biology, Lomonosov Moscow State University, Moscow, Russia; Faculty of Bioengineering and Bioinformatics, Lomonosov Moscow State University, Moscow, Russia*

NATASHA SHERSTNEV • *University of Dundee, Dundee, Scotland, UK; U. Paris Sud, Orsay, France*

FABIAN SIEVERS • *School of Medicine, University College Dublin, Dublin, Ireland*

BEN SOARES • *University of Dundee, Dundee, Scotland, UK*

DARON M. STANDLEY • *Research Institute for Microbial Diseases, Osaka University, Osaka, Japan; Immunology Frontier Research Center, Osaka University, Osaka, Japan; Graduate School of Medicine, Osaka University, Osaka, Japan*
DMITRY SUPLATOV • *Belozersky Institute of Physicochemical Biology, Lomonosov Moscow State University, Moscow, Russia*
VYTAS ŠVEDAS • *Belozersky Institute of Physicochemical Biology, Lomonosov Moscow State University, Moscow, Russia; Faculty of Bioengineering and Bioinformatics, Lomonosov Moscow State University, Moscow, Russia*
ERIC TANNIER • *Université de Lyon, Université Lyon 1, CNRS, Laboratoire de Biométrie et Biologie Evolutive UMR 5558, Villeurbanne, France; INRIA Grenoble-Rhône-Alpes, Montbonnot, France*
ANDRES VEIDENBERG • *Institute of Biotechnology, University of Helsinki, Helsinki, Finland*
TANDY WARNOW • *Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA*
SIMON WHELAN • *Department of Evolutionary Genetics (Program in Evolutionary Biology), Uppsala University, Uppsala, Sweden*

Part I

Basic Tools for Computing MSAa

Chapter 1

The Clustal Omega Multiple Alignment Package

Fabian Sievers and Desmond G. Higgins

Abstract

Clustal Omega is a version, completely rewritten and revised in 2011, of the widely used Clustal series of programs for multiple sequence alignment. It can deal with very large numbers (many tens of thousands) of DNA/RNA or protein sequences due to its use of the mBed algorithm for calculating guide-trees. This algorithm allows very large alignment problems to be tackled very quickly, even on personal computers. The accuracy of the program has been considerably improved over earlier Clustal programs, through the use of the HHalign method for aligning profile hidden Markov models. The program currently is used from the command-line or can be run online.

Key words Multiple sequence alignment, Progressive alignment, Protein sequences, Clustal

1 Introduction

Clustal Omega [1] is a package for performing fast and accurate multiple sequence alignments (MSAs) of potentially large numbers of protein or DNA/RNA sequences. It is the latest version of the popular and widely used Clustal MSA package [2, 3]. Clustal Omega retains the basic progressive alignment MSA approach of the older ClustalX and ClustalW implementations, where the order of alignments is determined by a so-called guide-tree [4], which in turn is constructed from pairwise distances among the sequences. The main improvements over ClustalW2 are (1) use of the mBed algorithm for creating guide-trees of any size [5] and (2) a very accurate profile-profile aligner, based on the HHalign package [6]. As a first step, a traditional progressive aligner calculates all $N(N\text{-}1)/2$ pairwise distances among all N input sequences. This may be computationally too demanding for more than 10,000 sequences. The mBed algorithm, as implemented in Clustal Omega, reduces the time and memory complexity for guide-tree calculation from $O(N\char`^2)$ to $O(N(\log(N))\char`^2)$. This is achieved by calculating the pairwise distances of all N sequences with respect to $\log(N)\char`^2$ randomly chosen seed sequences only. The fast pairwise distance calculation routines, based on a k-tuple alignment algorithm, have been retained from the previous Clustal programs. The pairwise distances are then clustered, using a bisecting *k*-means algorithm [7]. Groups of sequences are bisected until a certain threshold for the cluster size is reached. In the current version, this threshold is hard-wired to 100 but can be changed to larger or smaller values. Guide-tree construction within the clusters and among the clusters makes use of the tree building routines in Muscle [8]. This dendrogram is referred to as a guide-tree to emphasize that it is only used to guide the progressive alignment; it is not a reliable guide to the phylogeny of the sequences. Guide-tree construction will be skipped if only two sequences are to be aligned or if an externally constructed guide-tree is inputted.

In the profile-profile alignment phase, sequences are aligned in larger and larger groups, according to the branching order in the guide-tree. At each stage of this final step, two alignments are aligned. Initially these are single sequences, but they grow with the addition of new sequences as one traverses the guide-tree. The alignment of residues and the positioning of gaps during each profile-profile alignment are fixed and cannot be undone at a later profile-profile alignment higher up in the tree. The main algorithmic change over ClustalW2 is a new profile-profile alignment engine, based on the HHalign software [6]. HHalign is entirely based on hidden Markov methods (HMMs). Sequences and intermediary profiles are converted into HMMs, which are aligned in turn. It is also possible to input a HMM in addition to the unaligned sequences and to use this external HMM to guide the alignment during the profile-profile alignment stage. This is referred to as external profile alignment (EPA). There are two HMM alignment algorithms: the accurate and memory-hungry maximum accuracy (MAC) algorithm and the faster, less accurate, and more memory-efficient Viterbi algorithm. The MAC algorithm is the default, and Viterbi is activated automatically only if the system resources are exhausted.

Sequence input to Clustal Omega is handled by the Squid routines http://eddylab.org/software/squid/squid.tar.gz, and permissible input formats are a2m (fasta/vienna), clustal, msf, phylip, selex, and stockholm. Output can be in the same formats. The maximum number of sequences and lengths that can be aligned will depend on the machine being used. The number of sequences primarily affects the distance matrix calculation. Storing an mBed matrix for $N = 10{,}000$ sequences takes up approximately 14 MB of memory. A full distance matrix would take up almost 400 MB. Both alternatives are clearly feasible on a modern desktop computer. For $N = 100{,}000$ the mBed matrix will take up 220 MB, while the full distance matrix will require about 40GB, which may necessitate a higher-end machine. The length of the individual input sequences also contributes to the memory consumption of the tree-building phase via the *k*-tuple alignment distance calculation among pairs of sequences. The length of the final alignment, and therefore by extension the lengths of the input sequences, however, mostly impacts on the profile-profile alignment phase. For every profile-profile alignment, the MAC algorithm constructs six $L_1 \times L_2$ matrices of double variables, where L_1 and L_2 are the lengths of the two profiles to be aligned. An alignment of two profiles, each 100 residues in length, will therefore require $8 \times 6 \times 100 \times 100 = 480{,}000$ bytes. The maximum alignment length for a machine with 8GB would therefore be two profiles of $\sqrt{(8\ \mathrm{GB}/(6 \times 8)\mathrm{B})} = 13{,}377$ positions in length each, or equivalent (*see* **Note 1**). The number of sequences affects the resource requirements of the profile-profile alignment stage only indirectly, in that it influences how much the lengths of the intermediate profiles grow from the lengths of the individual sequences. This growth is difficult to predict and depends, among other factors, on the similarity of the sequences. The time required for the profile-profile alignment stage is a function of the number of sequences *N*, the lengths of the intermediate profiles *L*, and the shape of the guide-tree. An MSA of *N* sequences requires (*N*-1) profile-profile alignments; increasing the number of sequences increases the number of profile-profile alignments linearly, and therefore the alignment time will also grow in a linear fashion, at least in simple cases. Increasing the lengths of the input sequences clearly will increase the lengths of the intermediate profiles. Building up the HMM matrices requires a multiple of $L_1 \times {}_2$ operations, so increasing the lengths of the sequences will increase the matrix construction times in a quadratic fashion. The guide-tree topology affects the profile-profile alignment times in a subtle way. Roughly speaking, alignments generated using a balanced tree [9] will require less time than using an imbalanced (chained) tree [10]. For example, on a single core of a 64 bit 3.0GHz machine with 4GB of RAM, it takes just over 5 min to construct the tree and align 50,000 zinc-finger sequences of average length of 23 residues; it takes 25 min for 20,000 and 68 min for 50,000 sdr sequences of average length 163 residues. It takes 106 min for 20,000 p450 sequences of average length 331 amino acids.

The current implementation of Clustal Omega is command-line driven. There is as of yet no GUI and no interactive menu. A list of all permissible command-line arguments is available by typing –h (--help) on the command-line. There is an exhaustive help file explaining all command-line arguments and their usage in detail. The help file also contains many examples, showing the use of all individual command-line arguments and a range of typical combinations of command-line arguments.

2 Materials

Clustal Omega is available online for interactive use. Two sites offering Clustal Omega are http://www.ebi.ac.uk/Tools/msa/clustalo/ (*see* Fig. 1 and [11]) and https://galaxy.pasteur.fr/ (search for Clustal Omega).

Clustal Omega source code and executables can be downloaded from http://www.clustal.org/omega/.

Executables are provided for Linux (32/64 bit), Mac (64 bit), Windows (64 bit), and FreeBSD (32/64 bit). To compile Clustal Omega from source under Linux, one has to un-tar the distribution and change (cd) into the un-tar-ed directory, configure, and make. For example, if the tar-ball is called clustal-omega-1.2.4.tar.gz, then a typical installation might require:

```
$ tar -xvf clustal-omega-1.2.4.tar.gz
$ cd clustal-omega-1.2.4/
$ ./configure
$ make
$ make install
```

Do not type the “$”, this is the command-line prompt in a POSIX compliant shell; different operating systems and/or shells may have different command-line prompts, like “>” in Windows, “#” for a root user, or “%” in a C shell.

This should configure, build, and install the Clustal Omega package under Linux (*see* **Notes 2–4**). The last step may require root/sudo privileges. However, make will still compile a Clustal Omega executable, which can be moved to any location in the user’s file tree and can then be invoked by specifying the full pathname, for example:

```
$ /home/clustal-user/path/to/where/clustal/is/located/clustalo
```

For installation under Windows, *see* **Note 5**.

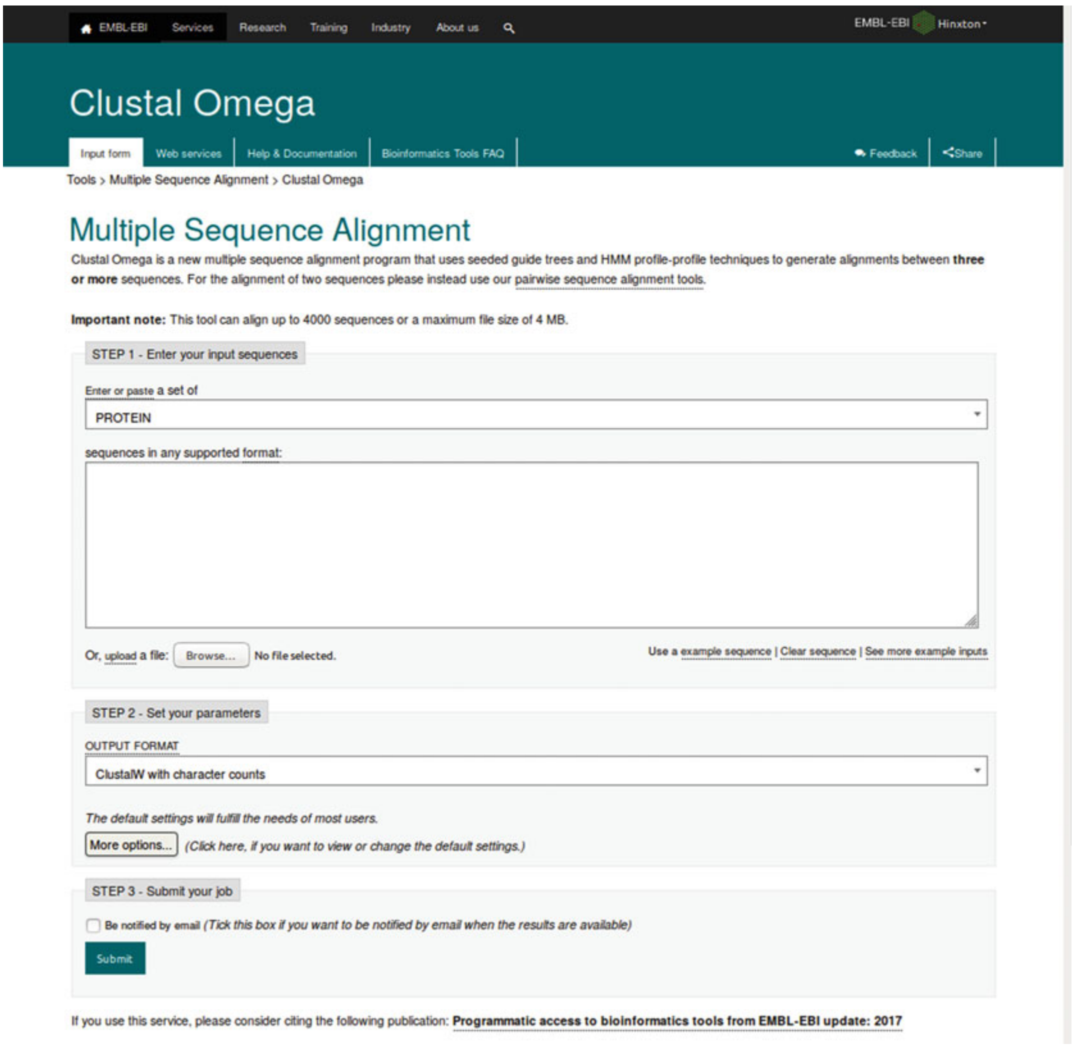
3 Methods

3.1 Basic Multiple Sequence Alignment

Clustal Omega can only be run locally from the command-line. To obtain a brief list of all available Clustal Omega command-line flags type:

```
$ clustalo -h
```

and hit <return>



Clustal Omega

Tools > Multiple Sequence Alignment > Clustal Omega

Multiple Sequence Alignment

Clustal Omega is a new multiple sequence alignment program that uses seeded guide trees and HMM profile-profile techniques to generate alignments between **three or more** sequences. For the alignment of two sequences please instead use our pairwise sequence alignment tools.

Important note: This tool can align up to 4000 sequences or a maximum file size of 4 MB.

STEP 1 - Enter your input sequences

Enter or paste a set of

PROTEIN

sequences in any supported format:

Or, upload a file: Browse... No file selected.

Use a example sequence | Clear sequence | See more example inputs

STEP 2 - Set your parameters

OUTPUT FORMAT

ClustalW with character counts

The default settings will fulfill the needs of most users.

More options... *(Click here, if you want to view or change the default settings.)*

STEP 3 - Submit your job

Be notified by email *(Tick this box if you want to be notified by email when the results are available)*

Submit

If you use this service, please consider citing the following publication: **Programmatic access to bioinformatics tools from EMBL-EBI update: 2017**

Fig. 1 Screenshot of the Clustal Omega Web page on the EBI Web. Site: http:/www.ebi.ac.uk/Tools/msa/clustalo/

Parts of the Clustal Omega code have been multi-threaded, using OpenMP. By default, Clustal Omega will attempt to use all available threads. To limit the number of threads one can specify --threads. This is recommended if Clustal Omega is not the only process running on the host machine.

Despite Clustal Omega being very fast for many purposes, some alignments may take a long time. In this case it may be reassuring to track the real-time progress of the alignment. This can be done by specifying the -v flag. This will print to screen what phase the calculation is in (distance matrix calculation, k-means clustering, guide-tree construction, progressive alignment). Repeating the –v flag a second time on the command-line increases the level of verbosity, giving a more detailed progress report. Triple -v is the highest level of verbosity, giving details about distances, tree building, and intermediate alignments. This level is only useful for the smallest alignments.

The most basic use of Clustal Omega involves aligning a number of unaligned sequences that are all contained in a single file. For example, if the file globin.fa contains two or more unaligned sequences in fasta format, then:

```
$ clustalo -i globin.fa
```

will read in the file, align the sequences, and output the alignment to screen (default) in the default (fasta) format.

```
$ clustalo -i globin.fa -o globin.sto --outfmt=st
```

If the file globin.sto does not exist, then Clustal Omega reads the sequence file globin.fa, aligns the sequences, and prints the result to globin.sto in Stockholm format. If the file globin.sto does exist already, then Clustal Omega terminates the alignment process before reading globin.fa.

```
$ clustalo -i globin.fa -o globin.aln --outfmt=clu --force
```

Clustal Omega reads the sequence file globin.fa, aligns the sequences and prints the alignment to globin.aln in Clustal format, overwriting the file globin.aln, if it already exists.

```
$ clustalo -i globin.fa --guidetree-out=globin.dnd --force
```

Clustal Omega reads the sequence file globin.fa, aligns the sequences, and prints the alignment to screen in fasta/a2m format (default) and the guide-tree to globin.dnd in Newick format, overwriting this file, if it already exists (*see* **Notes 6–8**).

```
$ clustalo -i globin.fa --guidetree-in=globin.dnd
```

Clustal Omega reads the files globin.fa and globin.dnd, skipping distance calculation, and guide-tree creation, using instead the guide-tree specified in globin.dnd. The inputted guide-tree must be in Newick format. The program will terminate if there is a discrepancy in sequence labels in the sequence and the guide-tree files. The alignment is outputted to screen in fasta format (default).

3.2 External Profile Alignment (EPA)

As mentioned in Subheading 1, Clustal Omega is a progressive aligner. This means that residues that are aligned and gaps that are positioned during an early stage of the alignment process remain fixed throughout the rest of the process and cannot be changed. An alignment of two residues that appears to be advantageous at an early stage could turn out to be suboptimal in the final alignment. External profile alignment (EPA) is a way to provide the alignment process with a certain degree of "foresight." If the final alignment can be anticipated, then this prior knowledge can be encoded as a HMM. This may be because the user knows they are aligning a particular type of sequence, for example, globins. Precomputed HMMs for globins are available from repositories such as Pfam (http://pfam.xfam.org). Alternatively, the user could have produced a manually curated high-quality alignment of sequences that are homologous to the input set. This alignment can then be converted into a HMM using, for example, HMMER [12].

Clustal Omega accepts these external profile-HMMs as input, accompanying the unaligned sequences. During the alignment stage, sequences and profiles are first aligned to the external profile, and pseudo-counts from the HMM are transferred to the internal HMM used to align the sequences progressively. The desired effect of this is to "nudge" particular residues and gaps toward the position where they are expected to end up in the final alignment.

Individual sequences and small profiles are most vulnerable to misalignment. On the other hand, large profiles have already built up more information about sequence variability at each position. Clustal Omega therefore up-weights the pseudo-count transfer for single sequences and intermediate alignments with small numbers of sequences and reduces the transfer to larger intermediate alignments. Pseudo-count transfer to alignments larger than, say, ten is negligible. Using EPA increases the profile-profile alignment time approximately threefold. Firstly, each of the two profiles is aligned to the external HMM, and finally the pre-aligned profiles have to be aligned themselves.

The most straightforward EPA scenario is where one background HMM is assumed to extend over the entire range of the final alignment. This is dealt with by the --hmm-in flag. To use a HMM in conjunction with unaligned sequences, first determine the appropriate HMM. For example, a search of the Pfam Web site for "globin" finds PF00042 which is the Pfam accession number for globins. Go to "Curation & model" of the PF00042 page and download the HMM called PF00042.hmm. Then type:

```
$ clustalo -i globin.fa --hmm-in=PF00042.hmm
```

However, if the unaligned sequences extend over multiple domains, and HMMs are only available for the individual domains, then one single HMM may not be capable of anticipating the entire final alignment. In this case multiple HMMs can be specified for individual sequences. This is achieved by specifying a HMM batch file in conjunction with the --hmm-batch flag. This batch file is a plain text file of the following format:

```
seq1 hmm1
seq2 hmm2
seq3 hmm3
… …
… …
```

where sequence label seq1 is associated with HMM file hmm1 and so on. This option is invoked as:

```
$ clustalo -i globin.fa --hmm-batch = globin-batch.txt
```

where globin-batch.txt is the HMM batch file.

3.3 Iteration

A useful means of refining an alignment is by "iterating" the alignment process. The guide-tree used for performing the initial alignment is based on pairwise distances between unaligned sequences. This may not be a reliable distance measure, and the guide-tree derived from these distances may not be ideal. A better distance measure between sequences is one based on a full multiple alignment [13]. In Clustal Omega such distances can be calculated from an initial multiple alignment and can be used to calculate a new, hopefully better, guide-tree. Any subsequent guide-tree refinement will again use the alignment distances between sequences. These distances are expected to become more accurate as the alignments they are based upon become more accurate, leading in turn to better guide-trees and by extension to better alignments. EPA required an externally computed HMM. This can be used to create a simple iteration scheme. In a first step unaligned sequences are aligned without any external profile. This produces an alignment which can be internally converted into a HMM and used in a second round of aligning in the same way as an EPA. Both of these steps, the initial unassisted alignment and the second alignment using a HMM and a new guide-tree derived from the first alignment, can be performed with one invocation of Clustal Omega:

```
$ clustalo -i globin.fa --iter=1
```

This will perform an initial alignment. It will then derive new distances between sequences from this alignment and construct a new guide-tree. It will also convert the initial alignment into a HMM and use this HMM in a second round of profile-profile alignment. After this second round, the final alignment is written to the screen. Guide-tree iteration and HMM iteration can be decoupled in the following way:

```
$ clustalo -i globin.fa --iter=5 --max-guidetree-iterations=1
```

This performs an initial alignment. In a second round (first iteration) one reconstruction of the guide-tree is performed in tandem with one profile-profile alignment using a HMM derived from the initial alignment. Four subsequent refinement rounds will use HMMs derived from the previous alignments but will not recalculate the guide-tree. Conversely, one can restrict the number of HMM iterations while repeatedly refining the guide-tree, by setting --max-hmm-iterations to a value less than the one specified by --iter. However, this variant is probably less useful, as it does not use any HMM information at the last alignment step (*see* **Note 9**). Good results have been achieved with --iter=1 and --iter=2; different iteration schemes should be used on a case-by-case basis.

If only the guide-tree and/or the distance matrix but no actual alignment are desired, then one can set --max-hmm-iterations=-1; this will skip the alignment phase.

3.4 Profile Alignment

When reading in aligned sequences, Clustal Omega makes use of the alignment information (full alignment distances for guide-tree construction and HMM information for EPA) and then "dealigns" the sequences (removes all gaps) before realigning them. If the --dealign flag is specified, then the sequences are dealigned without making use of the alignment information. Sometimes this is not desirable. For example, one might have a high-quality, hand-curated alignment to which some unaligned sequences are to be added while keeping the curated alignment fixed. Alternatively, one might want to align two profiles. In these cases one has to use the Clustal Omega --profile1 (and --profile2) flag. To align two profiles use (*see* **Note 10**):

```
$ clustalo --profile1=globin1.aln --profile2=globin2.aln
```

If more than one unaligned sequence, for example, in file moreGlobins.fa, are to be added to an existing profile use:

```
$ clustalo --profile1=globin1.aln -i moreGlobins.fa
```

Clustal Omega extracts HMM information from the profile and uses this HMM as an external profile for the alignment of the unaligned sequences. Once all the unaligned sequences in moreGlobins.fa have been aligned, this new profile is aligned to the previously existing profile globin1.aln.

4 Notes

1. By default, Clustal Omega uses the maximum accuracy (MAC) algorithm [6] to align profiles. This algorithm is memory intensive. If this amount of memory is not available, then Clustal Omega switches to the Viterbi [14] algorithm. However, it is not straightforward to automatically establish the amount of RAM available on different machines running different operating systems. Clustal Omega therefore assumes that it will have 8 GB = 8192 MB of memory available for the MAC alignment. Should one have more (or less) than 8 GB of RAM, then this can be communicated by setting the --MAC-RAM flag to the appropriate size (in MB). For example, on a machine with 4GB of RAM, one might specify.

   ```
   $ clustalo -i globin.fa --MAC-RAM=4096
   ```

   where 4096 corresponds to 4GB (=4 × 1024 MB). In practice one might want to reduce the --MAC-RAM value, to allow for memory usage other than the MAC alignment. If not enough RAM is available on the local machine but maximum accuracy is essential, a larger --MAC-RAM value can be specified. Clustal Omega will then access the machine's swap space. Swap space is often of the same size as RAM; however, this should be checked for the local installation. Accessing swap space is extremely slow.
2. The "configure" shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a "Makefile" in each directory of the package. It may also create one or more ".h" files containing system-dependent definitions. Finally, it creates a shell script "config.status" that one can run in the future to recreate the current configuration and a file config.log' containing compiler output (useful mainly for debugging configure).
3. Clustal Omega needs argtable2 (http://argtable.sourceforge.net/). If argtable2 is installed in a nonstandard directory, one may have to point configure to its installation directory. For example, if on a Mac with argtable installed via MacPorts, then one should use the following command-line:

   ```
   $ ./configure CFLAGS='-I/opt/local/include' LDFLAGS='-L/
   opt/local/lib'
   ```

4. Clustal Omega will automatically support multi-threading if the compiler supports OpenMP. For some reason automake's

OpenMP detection for Apple's gcc is broken. OpenMP detection can be forced by calling configure as follows:

```
$ ./configure OPENMP_CFLAGS='-fopenmp' CFLAGS='--
DHAVE_OPENMP'
```

5. For compiling and installing, Clustal Omega under Windows MinGW64 and MSYS should be installed. The compilation steps under MSYS are similar to the ones outlined under Subheading 2 (Materials). For a step-by-step guide, go to http://www.clustal.org/omega/INSTALL
6. Distance matrix output can be initiated by specifying --distmat-out. However, this is not possible in default (mBed) mode because mBed does not calculate a full distance matrix. Therefore one must specify full distance matrix calculation by setting

```
--full. For example:
$ clustalo -i globin.fa --distmat-out = globin.mat -full
```

   Distance matrix output in conjunction with iteration requires the --full-iter flag as well as the --full flag. If this flag is not specified, then Clustal Omega will write out the first distance matrix, based on k-tuple distances and will then perform the preliminary alignment. During the first iteration, it will then calculate new distances, based on the full multiple alignment, using the mBed algorithm. Since distance matrix output is not possible in mBed mode the preliminary matrix, based on k-tuples, will not be overwritten. Full alignment distances are used though for constructing an iterated guide-tree.
7. Guide-tree construction is based on pairwise distances. In this context fragments are particularly problematic. Fragments may be very short indeed and may therefore align perfectly with all sequences, leading to zero distances of all sequences with respect to the fragment. By transitivity, this insinuates that sequences that are in fact not close to each other can appear very close by proxy. This in turn will lead to a very bad guide-tree. This guide-tree will be extremely unbalanced (chained), which will lead to overly long execution times. It will also arrange the sequences in a more or less random order in this chained tree, leading to suboptimal alignments.
8. The guide-tree is built using Unweighted Pair Group Method with Arithmetic Mean (UPGMA). In UPGMA the nearest two clusters are combined into a higher-level cluster at each step. If there are ties, that is, more than one pair of clusters have the same nearest distance, then these ties are broken by the (random) order in which sequences appear in the input file. The

ordering of sequences in the input thereby may affect the shape of the guide-tree, which in turn may affect the final alignment [15].

9. A Clustal Omega iteration can be broken up into two distinct steps. Performing:

```
$ clustalo -i globin.fa -o globin-0.out
$ clustalo -i globin-0.out -o globin-1.out
```

is equivalent to

```
$ clustalo -i globin.fa --iter=1 -o globin-1.out
```

The first invocation produces an alignment called globin-0.out. During the second invocation Clustal Omega reads in globin-0.out and detects that this is a valid alignment. It does so by ascertaining that all input sequences have the same length and that at least one input sequence contains at least one gap. The alignment information is used to build a guide-tree from the full alignment distances (rather than from the k-tuple distances that were used during the first invocation), as well as to produce a HMM, which is used during the second profile-profile alignment stage. This approach may be desirable for certain reasons. Firstly, it retains the intermediate alignment, which is lost using the --iter flag. Secondly, it allows one to use (and refine) existing alignments which may have been produced by aligners other than Clustal Omega. For example, for moderate numbers of sequences, Kalign [16] is a faster alignment program than Clustal Omega while still producing alignments of reasonable quality.

```
$ kalign-2.04 -i globin.fa -o globin-0.out -q -f fasta
$ clustalo -i globin-0.out -o globin-1.out
```

This uses Kalign to create a rough but high-speed initial alignment, which is then refined using Clustal Omega. It is always advisable to ensure whether input sequences are actually aligned or not. In certain pipelines, unaligned sequences are arranged in such a way that sequences are padded at the end with gaps, such that all sequences have the same length. This is interpreted by Clustal Omega as a valid alignment, while in fact it is not. While the guide-tree that is derived from such an input is useless at best, the HMM information that is derived from this arrangement establishes the present, nonsensical, alignment. In this case one could either remove all gaps from the input by hand or specify the --dealign flag.

10. To align a single sequence to an existing profile, use the profile-profile syntax

```
$ clustalo --profile1=globin1.aln --profile2=singleSe-
quence.fa
```

When adding multiple sequences to a profile, Clustal Omega first aligns all the unaligned sequences, taking regard of the HMM information derived from the profile, and then aligns the newly formed profile to the already existing profile. If the profile/sequences mode were to be used for adding a single sequence, then Clustal Omega would complain because there is only one sequence during the first round of alignments, which cannot be aligned against any other sequence. Conversely, to add unaligned sequences one-by-one to an existing profile (rather than first aligning all the unaligned sequences and then aligning the new and the old profiles), one will have to distribute the unaligned sequences among multiple files and align the single sequences to the profile, overwriting the existing profile with the newly formed profile. One possible (bash) implementation to do this might be:

```
while read label; do
read seq;
echo $label > in.vie
echo $seq >> in.vie
clustalo --p1=globin-0.aln --p2=in.vie -o globin-0.aln --
force;
done < unaligned.vie
```

where unaligned.vie is the file that contains the unaligned sequences in Vienna format. Vienna format is the same as Fasta format but where all the residue information is in one (long) line – in this example Vienna format is mandatory. Globin-0.aln is the file that originally contains the existing profile. At every stage it is overwritten with the alignment comprising of the previous profile and one extra added sequence. It is advisable to keep a copy of the original alignment. When performing this procedure, the order in which unaligned sequences are added to the profile can impact on the final alignment.

References

1. 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

2. Higgins DG, Sharp PM (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene 73 (1):237–244

3. Larkin MA, Blackshields G, Brown NP et al (2007) Clustal W and Clustal X version 2.0. Bioinformatics 23:2947–2948
4. Higgins DG, Bleasby AJ, Fuchs R (1992) CLUSTAL V: improved software for multiple sequence alignment. Comput Appl Biosci 8 (2):189–191
5. 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
6. Söding J (2005) Protein homology detection by HMM-HMM comparison. Bioinformatics 21:951–960
7. Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms, Philadelphia, PA, pp. 1027–1035
8. Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32:1792–1797
9. Sievers F, Hughes GM, Higgins DG (2014) Systematic Exploration of Guide-Tree Topology Effects for Small Protein Alignments. BMC Bioinformatics 15:338
10. Boyce K, Sievers F, Higgins DG (2014) Simple chained guide trees give high-quality protein multiple sequence alignments. PNAS 111 (29):10556–10561
11. Chojnacki S, Cowley A, Lee J, Foix A, Lopez R (2017) Programmatic access to bioinformatics tools from EMBL-EBI update: 2017. Nucleic Acids Res 45(W1):W550–W553. https://doi.org/10.1093/nar/gkx273
12. Finn RD, Clements J, Eddy SR (2011) HMMER web server: interactive sequence similarity searching. Nucleic Acids Res 39(Suppl 2):W29–W37
13. Kimura M (1985) The neutral theory of molecular evolution. Cambridge University Press, Cambridge
14. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
15. Boyce K, Sievers F, Higgins DG (2015) Instability in progressive multiple sequence alignment algorithms. Algorithms Mol Biol 10:26
16. Lassmann T, Sonnhammer ELL (2005) Kalign - an accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics 6:298

Chapter 2

Phylogeny-Aware Alignment with PRANK and PAGAN

Ari Löytynoja

Abstract

Evolutionary analyses require sequence alignments that correctly represent evolutionary homology. Evolutionary homology and proteins' structural similarity are not the same and sequence alignments generated with methods designed for structural matching can be seriously misleading in comparative and phylogenetic analyses. The phylogeny-aware alignment algorithm implemented in the program PRANK has been shown to produce good alignments for evolutionary inferences. Unlike other alignment programs, PRANK makes use of phylogenetic information to distinguish alignment gaps caused by insertions or deletions and, thereafter, handles the two types of events differently. As a by-product of the correct handling of insertions and deletions, PRANK can provide the inferred ancestral sequences as a part of the output and mark the alignment gaps differently depending on their origin in insertion or deletion events. As the algorithm infers the evolutionary history of the sequences, PRANK can be sensitive to errors in the guide phylogeny and violations on the underlying assumptions about the origin and patterns of gaps. To mitigate the effects of such model violations, the phylogeny-aware alignment algorithm has been re-implemented in program PAGAN. By using sequence graphs, PAGAN can model and accumulate evidence from more complex gap structures than PRANK does, and incorporate this uncertainty in the inferred ancestral sequences. These issues are discussed in detail below and practical advice is provided for the use of PRANK and PAGAN in evolutionary analysis. The two software packages can be downloaded from http://wasabiapp.org/software.

Key words Phylogeny-aware alignment, Evolutionary sequence analysis, Insertions and deletions, Character homology

1 Introduction

Multiple sequence alignment has a central role in evolutionary sequence analysis, in some studies so central that the alignment and evolutionary inferences should be performed simultaneously or at least in a tightly coupled manner. The connection between alignment and phylogeny inference was noticed early [1] but the evolutionary consequences of it are still largely ignored by mainstream alignment methods. A probable explanation for this oversight is the historical focus on protein alignments and extensive use of structural benchmarks in the development and comparison of the analysis methods. The use of these benchmarks has produced great alignments for structural studies of proteins but, as noticed by many users of the resulting methods, the very same alignments may be unsuitable for evolutionary analyses.

This chapter focuses on evolutionary sequence alignment and the use of a phylogeny-aware alignment algorithm to infer alignments for evolutionary studies. The definition of evolutionary homology is central and we will start by discussing that and its correct representation in multiple sequence alignments. We will then introduce—with lots of figures and no equations—the phylogeny-aware alignment algorithm implemented in programs PRANK and PAGAN. After detailing the strengths and weaknesses of these methods, we will see what this means in practice and give advice for their use. We will finish with a brief discussion on the future plans for the phylogeny-aware alignment methods.

In the following sections, some methods based on the classical progressive algorithm are criticized and shown to perform poorly. This criticism is based on their performance in evolutionary analyses only and, as demonstrated in Chapter 17 of this book, the alignments they produce can be suitable for other types of analyses. Similarly, the phylogeny-aware algorithm may perform poorly in non-evolutionary alignment tasks and alternative methods should be used.

2 Evolutionary Homology in Sequence Alignment

A multiple sequence alignment represents site-wise homology or "identity" among the characters in different sequences. The type of homology/identity denoted by the alignment depends on the application and the intended use of the data: in evolutionary analyses, characters placed in the same alignments columns are believed to be evolutionarily homologous and share a common ancestor; on the other hand, many functional and structural analyses of proteins consider aligned characters positionally identical in proteins' tertiary structures. Evolutionary homology is not the same as structural and positional similarity and the difference between the two measures is most clearly evidenced by the role of insertions. Two independent insertions at the same position can lead to similar changes in the structure and the characters inserted independently may thus be considered structurally identical; in contrast, independent insertions—even at exactly the same position—do not share a common ancestor and can never be evolutionarily homologous. To correctly indicate the evolutionary homology of insertions, the characters descending from different insertions events should be placed in separate alignment columns (Fig. 1).

If one restricts the analysis to relatively short sequence fragments, one can assume that the sequences evolve by substitutions, insertions, and deletions only. The three processes can be assumed

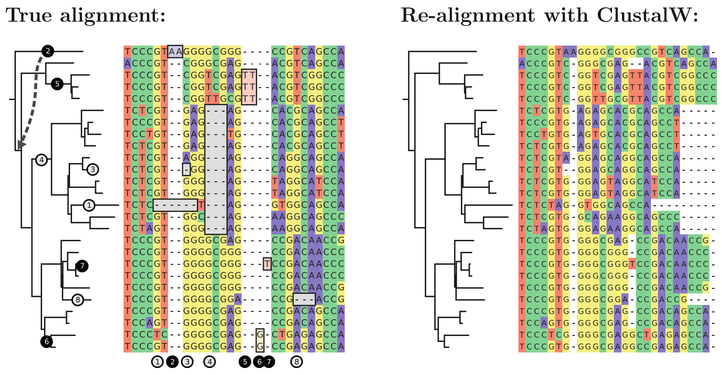


Fig. 1 The gap patterns in a true alignment reflect the underlying phylogeny of the sequences. Insertions and deletions create gap patterns (boxes in the alignment; numbered at the bottom) that reflect the phylogenetic locations of the events (black and gray bubbles in the tree). If multiple parallel insertions occur at homologous positions (events 5, 6, and 7), inserted columns can be placed in any order without effect on the homology statement. In the case of more than two parallel insertions, some inserted fragments are disconnected from the rest of the alignment (here, event 6). The type and phylogenetic location of event 2 are uncertain and it could also be a deletion in the sister branch (indicated by arrow). Whereas the phylogeny-aware algorithm would re-align the sequences correctly, the classical progressive algorithm (here ClustalW [2]) fails to resolve the true insertion and deletion events

to occur at relatively constant (although for different processes distinct) rates, substitutions typically being at least an order of magnitude more common than insertions and deletions [3]. The three processes differ greatly in their effect on the sequences and on one's ability infer the events from the data: (1) a character at a certain site can be substituted several times, subsequent substitutions may turn the character state back to an earlier one and characters in different evolutionary lineages may independently obtain the same new character state, all still remaining homologous to each other; (2) an insertion adds new characters to the sequence and subsequent insertions may be nested inside a fragment inserted by a preceding insertion event but, as mentioned above, insertions in different lineages are never homologous and evolve independently; and (3) a deletion removes characters permanently and the characters once deleted cannot be reverted, a potential insertion at the same position introducing new characters that are not homologous with the deleted ones.

By a comparison of two homologous sequences we can detect sites that have undergone substitutions but, without more information, we cannot tell which sequence has changed at which position. In contrast to this, length differences between two sequences can be explained by deletions of existing characters in one sequence or insertions of new characters in the other. The evolutionary lineage on which the substitution-differences between two sequences have occurred can be resolved using outgroup sequences and by inferring the character states for the ancestor of the two. Similarly to this, the evolutionary lineages—and thus the types—of insertion/deletion events creating length differences between two sequences can be inferred using information from phylogenetically related sequences.

Over time, sequences accumulate changes. Despite some differences in genome sizes, we can assume that sequences tend to retain their approximate length and insertion of new characters is counterbalanced by deletion of others. After a split from a common ancestor, the number of substitution-differences at homologous positions in descendant sequences increases until the sequence identity drops to the level expected by random sequences. The effect of insertions and deletions is very different. Assuming that each new insertion is not immediately followed by the deletion of the newly inserted characters, the total number of independent homologous sites within a set of sequence keeps increasing. With more than few sequences in the set, the increase in the number of independent homologous sites—and thus the number of columns in the alignment representing them—is not significantly affected by deletions as the chances of the same sites being independently deleted in all evolutionary lineages are small. Thus, the total length of the sequence alignment correctly representing the evolutionary homology among the characters is expected to grow roughly linearly with the evolutionary time covered by the different sequence lineages. Over long periods of time, the ancestral characters of a neutrally evolving sequence (or sequence region) are expected to be completely replaced by new characters through combinations of insertions and deletions: as a result, the correct evolutionary alignment of highly diverged descendant sequences should not match a single character. Typically, the more freely evolving sequence regions are flanked by conserved regions (e.g., loops and coils vs. core region in protein sequences) and the alignment is both possible and meaningful.

In practice, the alignment length rarely grows linearly with the evolutionary divergence. If the alignment is performed with methods based on the classical progressive algorithm [4, 5], the alignment length may grow linearly with the number of substitution changes for a while but the growth curves of the two then separate and the alignment length increases only slowly, if at all (Fig. 2). The reason for this is that the classical algorithm does not distinguish insertions from deletions and, inherently, considers all length differences as deletion events. The use of such biased alignments in evolutionary analysis is likely to lead to erroneous conclusions.

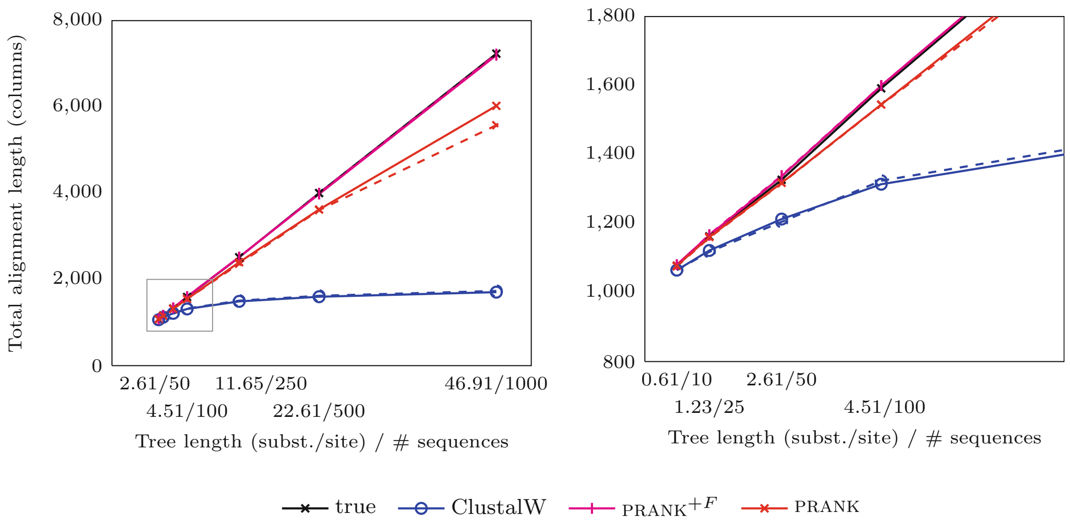


Fig. 2 The length of the alignment is expected to grow linearly with the evolutionary divergence contained within the sequences. One thousand sequences were simulated under a random tree with the maximum root-to-tip distance of 0.1 substitutions per site. Subsets of 10, 25, 50, 100, 250, and 500 sequences as well as the full datasets were re-aligned with ClustalW [2] and PRANK, and the length of the resulting alignments is plotted as a function of the length of the tree relating the included sequences. As the insertion–deletion process used for simulation is time-dependent and defined relative to the substitution rate, the correlation between the two values for the true alignment (black line) is perfect. The two variants of the phylogeny-aware function (PRANK and PRANK^{+F}) produce alignments with lengths close to the true length whereas a method based on the classical progressive alignment algorithm (ClustalW) over-aligns the sequences and the length of the alignment is seriously underestimated. Solid and dashed lines indicate alignments based on the true and estimated guide trees, respectively. The rectangle in the left plot indicates the area shown on the right

3 Phylogeny-Aware Alignment

Independent insertions at the same position are not homologous and have to be identified to allow for their correct placement in different alignment columns. This alone demonstrates that an evolutionarily accurate alignment cannot be generated without considering the phylogeny of the sequences included. In practice, not only are insertions at the same position problematic but the correct alignment of sequences with insertions and deletions at near-by positions requires the identification of distinct evolutionary events and their subsequent correct handling.

Progressive alignment algorithms exploit the sequence phylogeny and align the sequences pairwise in the reverse order, starting from the most closely related ones and, at each step clustering the aligned subsets, progressing towards the root of the tree. A major reason for the use of progressive algorithms is the prohibitive computational complexity of the exact multiple sequence alignment algorithm: with progressive algorithms, the complexity of aligning n sequences of length l is reduced from $O(l^n)$ to $O((n-1)l^2)$. The additional beauty of the approach is that the algorithm starts with alignments that are expected to be easiest and thus minimizes the chances of early alignment errors in its greedy processing of sequences. The classical algorithm does not use the phylogeny for anything else, however, and the placement of gaps—that is, the inference of which characters have been inserted or deleted in the evolutionary past—in the resulting alignments is often phylogenetically implausible [6].

Assuming that the alignment guide tree is correct and that the sequences are relatively closely related, the progressive alignment approach provides the information necessary to identify insertion and deletion events. The phylogeny-aware progressive algorithm implemented in programs PRANK [6, 7] and PAGAN [8] uses outgroup information from the next alignment step to decide if the length difference observed between the aligned sequences (representing either true extant sequences or internal nodes representing an aligned subset) was caused by an insertion or a deletion (Fig. 3). By identifying the true evolutionary event, the phylogeny-aware algorithm can handle insertions correctly and avoid penalizing the single event multiple times in later stages of the alignment.

The two methods are based on the same concept but they differ significantly on how they represent the uncertainty of the underlying cause of an observed length difference. PRANK is simpler in its design and flags the sites that contain an alignment gap in the immediately preceding stage of the progressive alignment, allowing for free placement of new gaps at flagged positions in the very next round. For an insertion, a new gap is created at exactly the same position and the flags indicating the gap are retained; for a gap caused by a deletion, a better alignment is obtained by matching the sites and the flags are removed (Fig. 3). PAGAN is a reimplementation of the phylogeny-aware algorithm using sequence graphs. Whereas alignment of two sequences with PRANK creates an ancestral sequence, alignment of two graphs creates an ancestral graph. This graph incorporates the edges of its descendants and, in the case of an alignment gap, contains two alternative paths across the sites (Fig. 3): the path through the unmatched sites represent a deletion in one descendant whereas the path skipping over the sites represents an insertion in the other descendant. The subsequent alignments can use either of these paths and thus resolve whether the length difference was caused by an insertion or a deletion. In both methods, the algorithm keeps the inserted sites at the later stages of the progressive alignment and the sequences (or graphs) it reconstructs for the internal nodes of the alignment tree may not reflect the true length of the ancestral sequences. Despite that, the identification and marking of the insertion events avoids penalizing for the same events multiple times and provides a significant improvement over the classical algorithm that, in practice, considers all length differences as deletions.

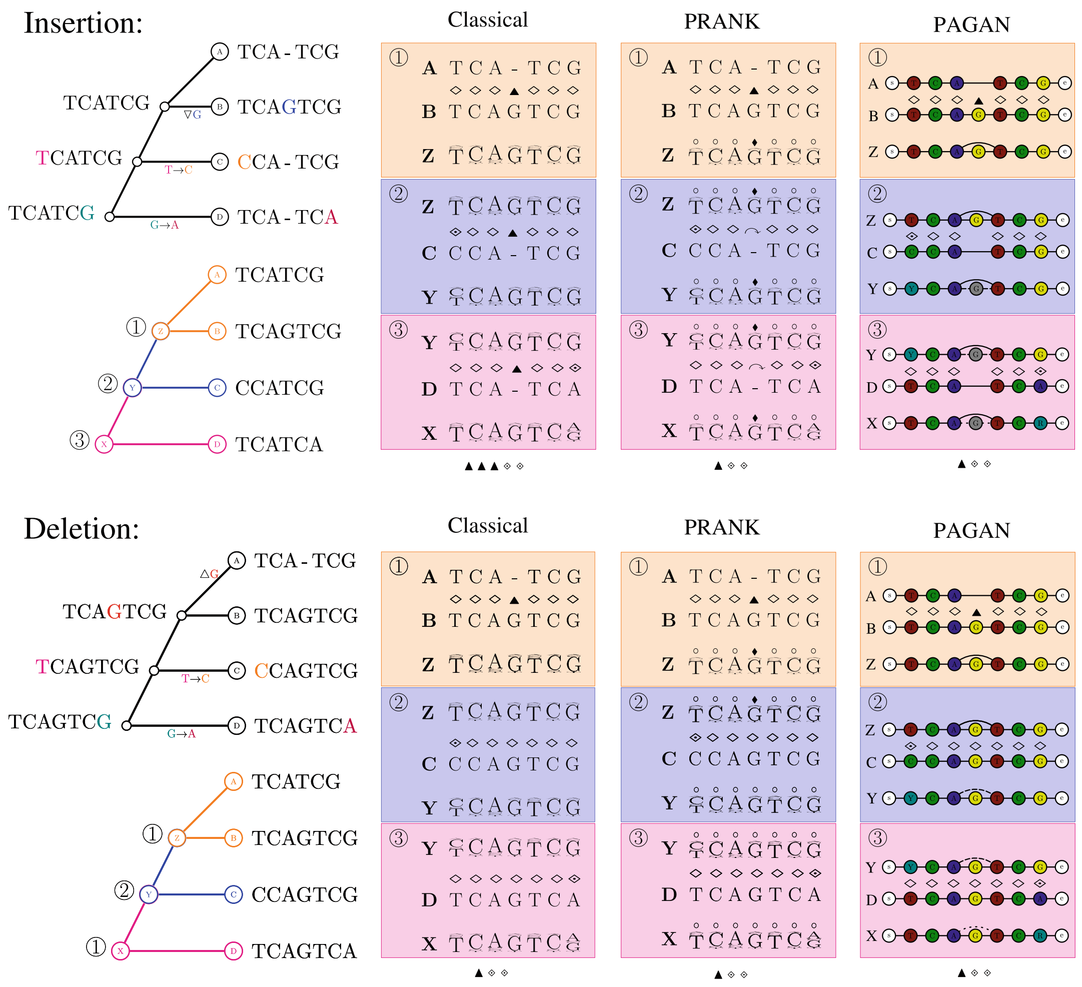


Fig. 3 The phylogeny-aware algorithm distinguishes insertions from deletions and treats them differently. The trees on the left represent the evolutionary histories of four short sequences undergoing two substitutions and either an insertion (top) or a deletion (bottom). The colored trees indicate how the alignment of sequences is divided into three pairwise alignments, each creating an ancestral sequence (**Z**, **Y**, **X**) that is placed at the corresponding internal node and then aligned pairwise with the next sequence. The classical alignment algorithm penalizes the single insertion three times (indicated with black triangle; diamond and diamond with dot denote match and mismatch, respectively); in contrast, the phylogeny-aware algorithm implemented in PRANK flags the gapped site after the first alignment (indicated by black lozenge above the sequence) and can then open a new gap at the flagged position without a further penalty (indicated by curved right arrow); PAGAN models sequences with graphs and represents alternative solutions with additional edges, adjusting the edge weights (shown with different line types) depending on their usage. For a deletion, the gap needs to be created only once and the phylogeny-aware algorithm either removes the flag indicating the gap after the second alignment or adjusts the weight of the unused edge. All methods produce the correct alignment

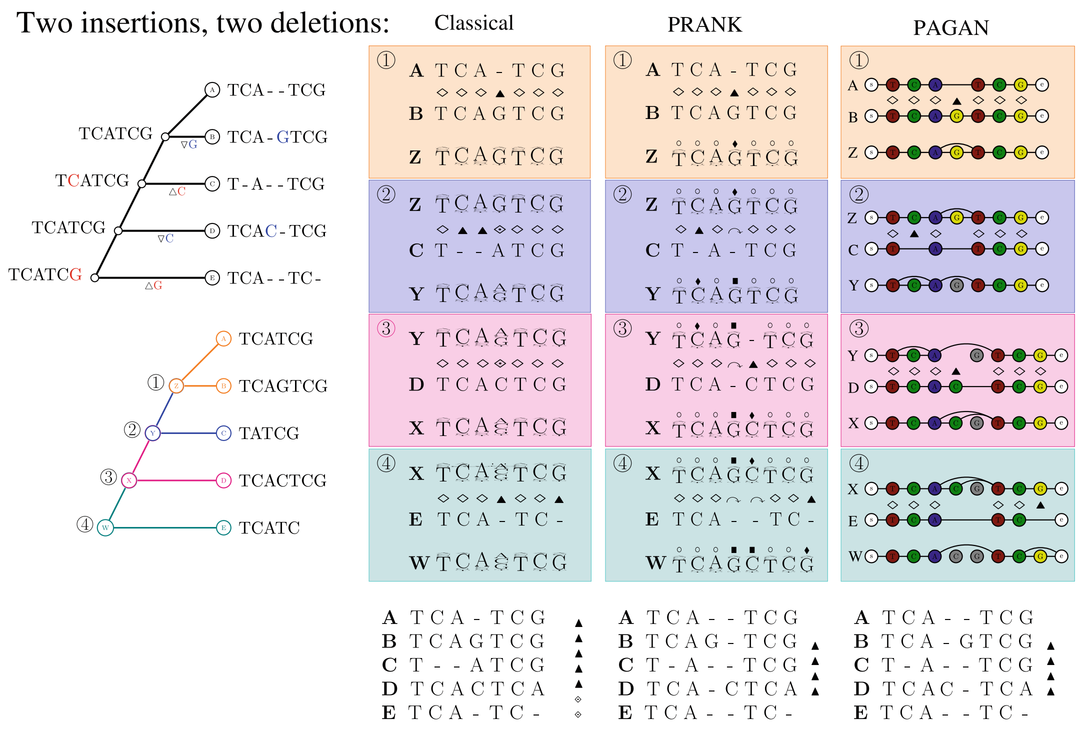


Fig. 4 The phylogeny-aware algorithm can distinguish and correctly align near-by insertion and deletions. The tree on the left represents the evolutionary history of five short sequences undergoing two insertion and two deletion events. The colored tree below indicates how the alignment is divided into pairwise alignments of sequences (or sequence graphs). The resulting alignments are shown on the bottom. The classical alignment algorithm considers length differences as deletions and cannot place independent insertions into separate columns; often it also moves near-by gaps and indicates false homologies, here resulting in substitutions. A variant of the phylogeny-aware algorithm with greedy calling of insertions, known as PRANK^{+F}, considers the re-use of a flagged gap as evidence that the gap was created by an insertion. It then changes the flags indicating a pre-existing gap (black lozenge) to ones indicating a permanent insertion (black square) and does not allow matching of these sites at later alignments. This forces the correct placement of independent insertions into separate alignment columns. The same functionality can be obtained with sequence graphs and greedy pruning of unused graph edges; this is not the default behavior of PAGAN, however. *See* Fig. 3 for the notation

Penalization of a single event multiple times seems an insignificant error if the procedure nevertheless reconstructs the correct alignment. In trivial alignment tasks that may be the case but in more complex ones the classical algorithm will allow for the matching of insertions with non-homologous characters, the resulting alignments indicating false homologies (Fig. 4). The heuristics proposed to correct for insertion events by lowering the gap cost at sites already containing gaps (e.g., [2, 9]) cannot prevent this; in contrast, they typically cause further errors by moving gaps caused by deletion events at near-by sites to the same columns and produce block-like alignments with alternating gappy and conserved regions (*see* Fig. 1). The basic version of the phylogeny-aware algorithm greatly reduces the problem but even that cannot completely avoid the matching of independent insertions, especially in the alignment of large datasets in which the chances of mutation events at near-by positions is significant (Fig. 2).

As discussed above, the phylogeny-aware algorithm identifies the type of insertion–deletion event and then handles the event accordingly, either creating a new gap or removing the flags indicating the gap. A variant of the algorithm, known as PRANK^{+F}, uses this information to mark sites at which the flagged gap is re-used as permanent insertions that cannot be matched at the later stages of the progressive alignment; to prevent overlapping deletions from confirming embedded insertions, the re-use of a gap has to be done for its full length with matching characters at the flanking sites. This approach can separate multiple insertions at the same position to independent events without effect on the placement of gaps caused by deletions (Fig. 4). When the order of aligning the sequences is correct and the sequence sampling is dense enough to call near-by gaps as separate events, PRANK^{+F} works very well and can reconstruct alignments with lengths very close to the true length (Fig. 2). When the underlying assumptions hold, the method in principle scales up to any number of sequences.

The phylogeny-aware algorithm reconstructs ancestral sequences with information about sites that are believed to be insertions. The ancestral sequences are required for the alignment but they can be useful otherwise, too: PRANK allows for outputting inferred ancestral sequences, using gaps to indicate sites that are believed to have been later inserted and not present in the ancestors, along with the alignment of the extant sequences. Such alignments are unique and enable studying the process of change and timing certain events to specific evolutionary branches. In addition to ancestral sequences, the algorithm also infers the type of mutation events that have caused the length differences between the sequences and can provide this information in the output. Although an experienced user may distinguish insertions and deletions from the gap patterns they create, the marking of gaps caused by insertions and deletions with different symbols, as can be done with PRANK, is helpful.

The explanation and illustration of the flagging approach used by PRANK is slightly simplified and only considers one level of flagging. In practice, the algorithm marks the gaps in the immediately preceding alignments and, for the sites not cleared of flags, for the one before that. This procedure prevents long deletions in one branch from masking overlapping insertions in the descendants of its sister branch. For details, *see* [6, 7].

4 Limitations of the Phylogeny-Aware Algorithm in PRANK

Unlike typical progressive alignment algorithms, the phylogeny-aware algorithm does not align sub-alignments to each other but reconstructs ancestral sequences to represent the parents of sets of aligned descendant sequences and then aligns pairwise these ancestral sequences. Accurate representation of the ancestral sequences, including the detection of inserted and deleted sites, is required for the correct distinction between insertion and deletion events in the subsequent stages of alignment. Correct reconstruction of sequences naturally requires that such ancestral sequences really existed and were true ancestors for the given sets of descendant sequences.

As the ancestral sequences are reconstructed for the internal nodes of the alignment phylogeny, it is crucial that the phylogeny accurately reflects the evolutionary history of the sequences. The role of alignment phylogeny is especially central in the calling of permanent insertions (PRANK^{+F}) that considers the re-use of a flagged gap as a confirmation that the gap has been created by an insertion. With the wrong order of aligning the sequences, a deletion may appear as an insertion and, by marking sites incorrectly as a permanent insertion, the algorithm has to place characters truly homologous to that to separate columns (Fig. 5). Although the resulting alignment is too long and gappy, small errors in the alignment order may not be too serious in typical evolutionary analyses: an incorrect alignment such as that in Fig. 5 does not indicate all true homologies but neither does it contain false homology statements.

In addition to the wrong alignment order, missing data can cause errors with the PRANK^{+F} variant. The algorithm assumes that alignment gaps are caused by insertions and deletions and then chooses the most plausible explanation of the two. One isolated gap caused by missing data may not be serious but if several sequences lack data at the same region, the gap pattern created may look like an insertion in the complete sequences; when this region is falsely marked as a permanent insertion, the subsequent alignment must place the affected region in separate columns. As sequences are often truncated at their ends, the marking of terminal gaps as permanent insertions is by default disabled by PRANK. Similar heuristics unfortunately cannot be provided for missing data in other parts of the sequences.

The phylogeny-aware alignment algorithm assumes that each alignment gap is caused by one insertion or deletion event and that the very next alignment provides information to distinguish between the two types of events. When the sequences are relatively closely related (and, as stated previously, the alignment order is correct), these assumptions are typically valid. If the sequences are

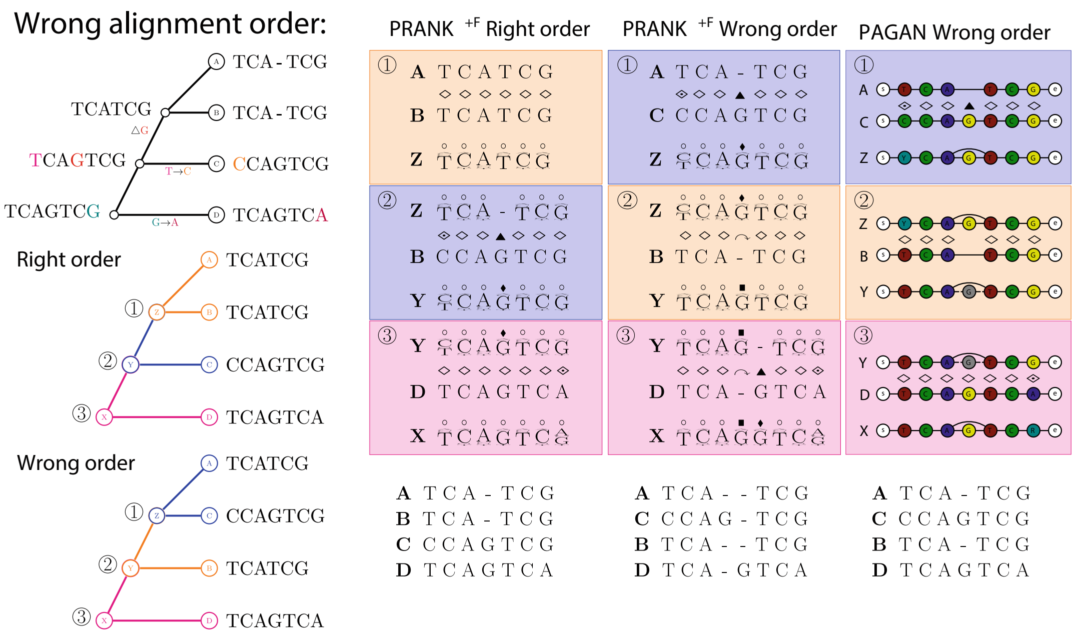


Fig. 5 The phylogeny-aware algorithm can be sensitive to errors in the guide phylogeny. The tree on the left represents the true evolutionary history and the colored trees below indicate the right and a wrong order of aligning the sequences. The greedy calling of insertions (PRANK^{+F}) marks flagged gaps that are re-used (curved right arrow) as permanent insertions (black square). When the alignment order is correct (left column), the algorithm works perfectly. If **A** and **C** are incorrectly aligned first (middle column), the subsequent alignment of **B** appears to confirm an insertion in **C** although the true event is a deletion shared by **A** and **B**. As the insertion in column 4 is marked permanent (black square), the site belonging to that column has to be placed in a column of its own. The resulting alignment is too long and gappy. PRANK (without $^{+F}$; not shown) and PAGAN (right column) are not affected by this error in the guide phylogeny and produce the correct alignment. *See* Fig. 3 for the notation

more diverged, the chances of independent insertion and deletions events at near-by positions in the adjacent evolutionary branches become significant. As a result of this, either the gap created in the first alignment may be a combination of two or more separate events or the subsequent alignment of an outgroup sequence fails to confirm the event as an insertion or a deletion due to an overlapping independent event in the neighboring branch (Fig. 6).

Some of the limitations of the approach and the measures to overcome them can be contradicting. Accurate calling of insertion and deletion events requires densely sampled sequence sets but the inference of alignment phylogeny for a large dataset is prone to errors [10] and the alignment may therefore suffer. Furthermore, incomplete lineage sorting is more likely among closely related sequences and possibly no single phylogeny correctly reflects the evolutionary history of all sites of a very densely sampled sequence set.

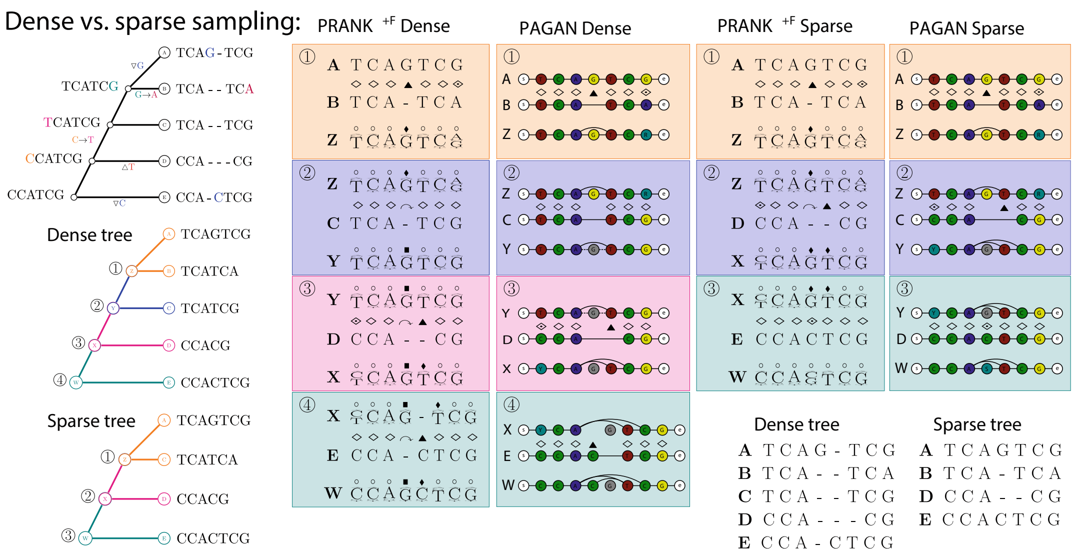


Fig. 6 Correct identification of independent insertion and deletion events requires closely related sequences. The tree on the left represents the true evolutionary history and the colored trees below indicate dense and sparse sampling of sequences. With dense sampling of sequences (middle tree) each insertion and deletion event can be identified using the information from the next alignment and the correct homology is recovered. With sparse sampling (bottom tree), the insertion in **A** cannot be identified because of a deletion at an adjacent position in **D**. As a result, the independent insertions in **A** and **E** are incorrectly matched. PRANK^{+F} and PAGAN are similarly affected by the sequence sampling; although the two methods place the insertions in different order, the resulting alignments are effectively the same

5 Phylogeny-Aware Alignment of Sequence Graphs with PAGAN

The idea of using partial order graphs for sequence alignment is old [11] but its implementation in PAGAN is novel. In PAGAN, the graph nodes represent sequence sites, for which the ancestral states are reconstructed using parsimony, while the graph edges indicate the possible paths for the alignment. A crucial feature of PAGAN's graph edges are the unequal weights that depend on the usage of the edges in the previous steps of the alignment. If certain edges are repeatedly not used, the sites (i.e., graph nodes) that they lead through are likely to be an insertion in one of the descendant sequences. As enough evidence for this is gathered (represented by dashed lines in Fig. 3, top), these edges are completely removed and subsequent matching of the sites is disabled. This behavior is very similar to permanent insertions of PRANK^{+F} but the implementation in PAGAN allows for more flexibility: each edge has a weight and the decision to call something an insertion and remove the flanking edges can be based, e.g., on the number of alignments or the phylogenetic distance since the edges were last used. Figure 4 shows greedy pruning of unused edges after only one alignment (similar to option + F in PRANK) while the alignments in Fig. 6 require evidence from two additional sequences.

PAGAN's more flexible modelling and usage of phylogenetic information in the calling of insertions and deletions fixes many shortcomings in PRANK but even it cannot do miracles. PAGAN is less sensitive to errors in the alignment order and the phylogenetic inconsistency (e.g., due to incomplete lineage sorting) between different gaps than PRANK^{+F} (Fig. 5). Similar to PRANK^{+F} (but not necessarily PRANK), PAGAN can align independent insertions at the same position into separate columns (cf. Fig. 1). However, both methods require information from which to call the gap type and also PAGAN does better with large numbers of densely sampled sequences than with few sparsely sampled ones (Fig. 6). PAGAN was designed to be faster and, instead of likelihood-based reconstruction of ancestral sites of PRANK, it uses maximum parsimony (cf. Fig. 3). Furthermore, PAGAN uses pre-computed scoring matrices, calculated for each alignment step from an evolutionary substitution model using the phylogenetic distances between the sequences, and represents each sequence position, even in the case of ambiguity, with one symbol. In the case of DNA, the standard ambiguity code is applied but for protein and codon sequences PAGAN can only represent ambiguity between two states and, for more complex cases, uses generic wildcards (X and NNN) that match equally well any other character. Although faster, this simplification can have serious downsides and PAGAN is likely to perform poorly in the alignment of a small number of highly diverged sequences. It can perform very well, however, when these highly diverged sequences are accompanied by many other sequences and the sequence sampling is both dense and even [12].

Although PAGAN can nowadays do de novo multiple sequence alignment, it was initially designed—and still best supported—for phylogeny-aware alignment extension. Here, "alignment extension" means addition of new sequences into an existing multiple sequence alignment such that the relative alignment of the original sequences is not changed. PAGAN does this in the context of the sequence phylogeny by removing a subtree, aligning sequence(s) to that and then grafting the extended subtree back to its original place in the full phylogeny. The new version of the program, PAGAN2, can do this in genomic scale and can generate and extend alignments of closely related sequences that are up to millions of bases long. PAGAN can also model uncertainties in the input data and has built-in support for the high error rate of homopolymers tracts in pyrosequencing data and for the high insertion–deletion error rate of the latest generation long-read sequencing data.

6 Phylogeny-Aware Alignment in Phylogenetic Analyses

PRANK and PAGAN are more "phylogenetic" than many alternative sequence aligners and aim to reconstruct the sequence history with accurate ancestral sequences. However, they are not true phylogenetic algorithms in the sense of, e.g., BAli-Phy [13], and are based on a progressive heuristic. The term "phylogeny-aware" refers to their use of phylogenetic information to distinguish insertions from deletions and ability to make gap patterns that correctly reflect the guide phylogeny. With all these phylogeny-related words, it may be surprising that the two programs should be used with caution in phylogenetic analyses.

As discussed above, PRANK and PAGAN create gap patterns that reflect the phylogenetic locations of insertion and deletion events (c. Fig. 1). In order to do that, they need information about the phylogenetic relationships of the sequences and obtain that from the alignment guide tree. It is important to understand the two methods create these phylogenetic gap patterns by the design of the algorithm and force the patterns to match the guide tree even when the guide tree is incorrect and does not reflect the true phylogenetic relationships of the sequences or the true pattern of gaps. It is unclear how specific errors created by the incorrect guide tree affect subsequent phylogenetic analyses but the errors are likely to be less random—and thus possibly more harmful—than the errors created by "non-phylogeny-aware" methods.

We studied the performance of PRANK and PAGAN in phylogenetic analyses and compared them to heuristics that iterate the alignment and phylogeny inference steps [14]. We found that when the sequence evolution perfectly matches the guide tree, PRANK-generated alignments are very accurate and the phylogenetic trees estimated from them can be more accurate than the trees estimated from true alignments (Fig. 7, left). The result may sound positive but is actually slightly alarming and indicates that the PRANK-made errors are biased towards the alignment guide tree. We also saw that both PRANK and PAGAN improved with denser sequence sampling and produced more correct alignments for 400 sequences than for a subset of 50 sequences; consistent with this, the phylogenetic trees estimated from the large alignments were more accurate (Fig. 7, middle). However, we also found that the alignment accuracy and the phylogenetic accuracy, as measured by the proportion of matching character pairs [15] and topological distance [16], did not correlate: despite their greater alignment error, the SATé-generated alignments produced in average more correct phylogenetic trees than the PRANK- and PAGAN-generated alignments (Fig. 7, bottom).

SATé is an iterative method that repeats the alignment and tree inference steps, in this case with MAFFT [17] and RAxML [18],

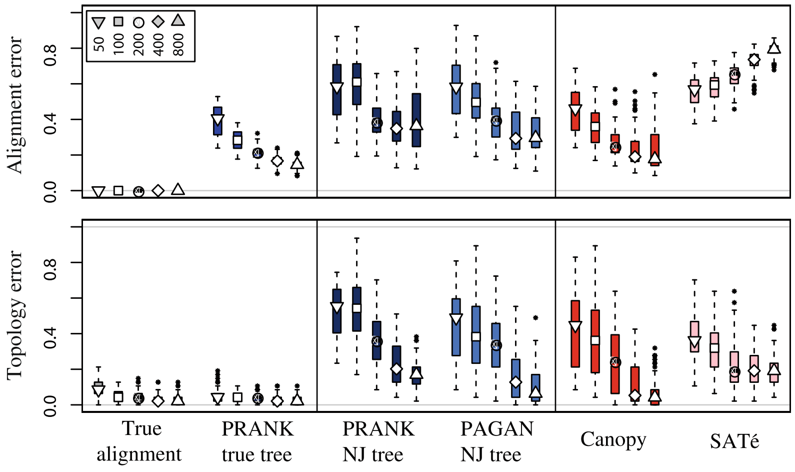


Fig. 7 PRANK and PAGAN produce better alignments with dense sampling of sequences. Simulated data were aligned with PRANK, PAGAN and two iterative approaches [14], and the accuracy of the alignments (top) and the phylogenetic trees inferred from them (bottom) were evaluated. Alignment accuracies of all phylogeny-aware methods improved with denser sampling of sequences while that of SATé decreased. Although similar improvements were seen in topological accuracies, the correlation was not perfect and in many cases SATé produced the most correct trees. For the very largest datasets, Canopy produced both the most accurate alignment and the most accurate phylogenetic tree, closely followed by PAGAN. The bottom row indicates the type of guide tree used in PRANK and PAGAN alignments; all phylogenetic trees were inferred with RAxML

several times. We developed a similar iterative approach, called Canopy, for phylogeny-aware algorithms and studied if the iteration reduces their guide tree-related error and thus improves their alignment and phylogenetic accuracy. This was indeed the case: Canopy (iterating PRANK and RAxML) produced more accurate alignments than the three other methods (Fig. 7, top; see also Fig. 3 in [14]) and more accurate phylogenetic trees than one round of PRANK and RAxML. However, the Canopy-generated trees were more accurate than the SATé-generated trees only for the large datasets and the trees estimated from the small datasets, despite the datasets being more accurately aligned, were worse than those generated by SATé. On the other hand, one round of PAGAN and RAxML produced results comparable to those of Canopy, suggesting that the new implementation of the phylogeny-aware algorithm is indeed less affected by small errors in the alignment guide phylogeny.

The take-home messages from this are somewhat mixed. The phylogeny-aware alignment methods can be used in phylogenetic analyses and they can do very well. However, they have large variance and can do very well, rather poorly or anything between; the iterative approach using a classical alignment algorithm seemed more robust in its performance. The positive finding is that, perfectly consistent with the theoretical background explained above, PRANK and PAGAN improve their performance with denser sampling of sequences. A concrete consequence of this exercise is that the latest version of PAGAN picks the best approaches from each step and first computes an alignment with MAFFT, then estimates a tree with FastTree [19] and finally performs the phylogeny-aware alignment based on that tree. A similar iterative approach can, of course, be done manually with PRANK or using Canopy. Finally, the analysis demonstrates the poor correlation between the accuracy of the alignment and the accuracy of the phylogenetic tree inferred from that. In line with this, modern alignment method comparisons are measuring the performance of the alignments in downstream analyses, not the column-wise accuracy of the alignment itself.

7 Practical Advice for the Use of PRANK

Evolutionary sequence analysis is based entirely on multiple sequence alignment and the accuracy of the downstream analysis depends on the correctness of the underlying alignment. Alignments produced with PRANK have been shown to provide accurate inferences of selection on protein-coding sequences [20, 21] and of ancestral sequences [12], and perform well in phylogenetic analyses [22], although the last finding is somewhat controversial due to the role of the guide phylogeny in the phylogeny-aware alignment. Despite its good performance in evolutionary analyses, PRANK is sensitive to violations of the assumptions made by the algorithm and the users of the program should understand the requirements and limitations of the method.

Alignment Phylogeny: *The phylogeny-aware alignment algorithm uses the alignment guide phylogeny to distinguish insertions from deletions. The algorithm is therefore sensitive to errors in the guide phylogeny, the variant with permanent insertions (PRANK^{+F}) being especially sensitive. Any PRANK alignment should be performed using an accurate guide phylogeny: if a high-quality phylogeny is available for the sequence set, it should be used instead of the heuristic phylogeny inferred by the program. In phylogenetic analyses iterative approaches similar to Canopy [14] are recommended.*

Evolutionary Distances: *PRANK uses the branch lengths provided by the guide phylogeny to re-compute the substitution and gap scoring for each alignment step. Depending on the expected evolutionary divergence, a region with several dissimilarities may be considered homologous and matched (distant sequences), or non-homologous and placed in separate columns (close sequences). Although the algorithm is not sensitive to small deviations in the branch lengths provided, a* *guide phylogeny with accurate distance estimates should be used when available.*

Option +F: *Given that the alignment guide phylogeny is correct and the sequence sampling is dense, the variant with permanent insertions (PRANK^{+F}) has been shown to outperform the basic algorithm [6]. If the alignment guide phylogeny is likely to contain errors or the input sequences are incomplete (i.e., contain missing data), the option +F can be problematic and the resulting alignment should at least be compared to one produced without it.*

Reproducibility: *Most pairwise alignments have several equally good solutions. In progressive alignment, the choice between these alternative solutions may trigger larger changes in the later stages of the process and lead to very different multiple alignments. Most alignment methods are deterministic and always pick the same solution and thus guarantee to produce the same final alignment. This practice hides the uncertainty in the data and has led to post-processing methods to recover the hidden variation [23]. By default, PRANK picks randomly one of the alternative solutions and may produce different results on independent runs of the very same data. This behavior may be disabled if reproducibility is required. By default PRANK iterates the alignment and the tree inference steps (using the neighbor-joining algorithm for the latter) and keeps the solution that has the best parsimony score. Unlike typical parsimony scores, the PRANK score considers substitutions as well as insertions and deletions.*

Sequence Alphabet: *PRANK represents sites at ancestral sequences with vectors of conditional likelihoods for the descendant subtree given different character states at the parent. This requires* $\mathrm{O}(\mathrm{A}^2)$ *computations for each cell in the dynamic programming matrix, where* A *is the size of the character alphabet, and makes the alignment of sequences with a large alphabet relatively slow. For protein-coding sequences, the alignments performed on codon level has been shown to outperform those done on protein sequences [20, 21]. Despite its slower computation, the use of codon alignment is recommended whenever possible. In general, protein-coding DNA sequences should not be aligned as DNA without good reason. If codon alignment is found to be too slow, PRANK provides an option to translate protein-coding DNA sequences to protein, perform the alignment on protein sequences and back-translate the resulting alignment to DNA.*

Sequence Sampling: *Given that the alignment guide phylogeny is correct and the sequence sampling is dense, PRANK is unbiased and scales up to any number of sequences. Even if the question in hand would not require an alignment of a large number of sequences, the quality of the resulting alignment is expected to be better when it is performed for many closely related sequences than for a small number* *of distantly related ones. Unneeded sequences can be removed after the alignment without affecting the statement of homology among the remaining sequences. PRANK is not suitable for the alignment of highly diverged sequences.*

8 Practical Advice for the Use of PAGAN

PAGAN was meant to replace PRANK but this has not really happened. One reason is that some decisions originally taken to speed up the alignment have compromised the program's performance and PAGAN currently does poorly in the alignment of very distant protein sequences. Furthermore, the work on PAGAN has led to unpredicted but still very interesting directions and the program has found a niche of its own. As PRANK and PAGAN are based on the same concept, their requirements and limitations are very similar.

Alignment Phylogeny: *PAGAN is less sensitive to errors in the guide tree than PRANK. However, the whole concept of phylogeny-aware alignment is based on the usage of information provided by phylogenetically related sequences and as good a guide tree as possible should be used. By default PAGAN uses FastTree to estimate the guide tree.*

Evolutionary Distances: *PAGAN recomputes the scoring matrix for each alignment step from an evolutionary substitution model. The scoring matrix is based on the branch length provided by the guide tree and therefore the branch lengths do matter. Similarly to PRANK, the scoring matrix is rarely critical for the alignment as long as the distances are of correct magnitude.*

Edge Pruning: *PAGAN has no option +F but it does a similar trick by removing unused edges. The rules for adjusting the edge weights or removing edges are not thoroughly tested and users are advised to experiment with the parameters if they are unhappy with the result produced with the default parameters. PAGAN can also be used for pileup alignment where sequences are simply added in the order of appearance as if they would be related by a ladder-like tree. With that, the use of option "keep-all-edges" may be useful as it allows reusing edges even when related sequences are not consecutive in the input file.*

Sequence Alphabet: *PAGAN was designed speed in mind and, although it uses an evolutionary model to compute the scoring matrix, its representation of ancestral sequences is simplistic, especially for amino acids and codons. PAGAN represents ambiguous nucleotides with the standard ambiguity code but, for computational reason it cannot do the same for amino acids and codons. This is not serious in the alignment of rather closely related sequences and for those PAGAN is really fast: there is some overhead from the computation of the scoring matrices for larger alphabets, but after that the data type* *makes no difference and the alignment of DNA and protein sequences is equally fast.*

Sequence Sampling: *As for PRANK, PAGAN requires (and can efficiently utilize) densely sampled sequence and is not suitable for the alignment of highly diverged sequences.*

9 Future Directions

PRANK has been shown to perform well in benchmarks assessing the suitability of sequence alignments generated with various methods to different types of evolutionary analyses [12, 20–22]. Despite the use of PRANK in many phylogenetic analyses, both PRANK and PAGAN should be used with caution in analyses where the guide phylogeny is unknown prior to alignment. On the other hand, if the problem with the guide phylogeny can be sorted out, the methods are expected to provide superior alignments for evolutionary analyses, often closely approximating alignments produced with computationally much heavier statistical methods [13, 24].

Although an iterative search strategy should help PRANK to greatly reduce the problems caused by an incorrect start guide phylogeny, iteration does not decrease the greediness of the algorithm nor can it solve the phylogeny for datasets that have no unique phylogeny, e.g., due to incomplete lineage sorting. These were the reasons to start developing PAGAN and to re-implement the phylogeny-aware algorithm for the alignment sequence graphs [8]. By using additional edges to indicate unresolved gaps and then pruning the unused edges after the alignment of related sequences, one can implement an algorithm very similar to that of PRANK^{+F} (Fig. 4). The advantages of the graph approach are greater, though, and instead of greedily pruning the edges, they can be given weights or probabilities based on the evidence for the different mutation types. Such a flexible edge-weighting makes PAGAN far less sensitive to errors in the guide phylogeny or different sites evolving under slightly different phylogenies while still allowing for correct separation of independent insertions into columns of their own.

As discussed above and evidenced by methods for co- and joint-estimation of alignment and phylogeny, the multiple sequence alignment should always be seen with the associated phylogeny. Understanding the alignment and drawing right conclusions from it is much easier when the relationships between the sequences are indicated next to the alignment. For methods such as PRANK and PAGAN, the phylogeny is also needed to indicate the relative positions of the ancestral sequences and to visualize the changes happening in different evolutionary branches. For this, we have developed Wasabi, a browser-based graphical user interface, that integrates these ideas and provides an easy-to-use access to PRANK, PAGAN, and many other evolutionary analysis methods [25]. The Wasabi server is available at http://wasabiapp.org/ and the use of PRANK and PAGAN within the Wasabi environment is described in detail in Chapter 14 of this book. The two software can be downloaded from the program home pages at http://wasabiapp.org/software. Pre-compiled packages of PRANK are provided for Linux, OSX, and Windows while the latest version of PAGAN is currently available for Linux only. PRANK and PAGAN are also provided as minimal Docker images that can be run on all platforms. Instructions for installation and usage of different versions are given on the program home pages.COMP: Please provide cross link for "Chapter 14" in the sentence "The Wasabi server ..."

References

1. Sankoff D (1975) Minimal mutation trees of sequences. SIAM J Appl Math 28:35–42
2. Larkin M, Blackshields G, Brown N, Chenna R, McGettigan P, McWilliam H, Valentin F, Wallace I, Wilm A, Lopez R, Thompson J, Gibson T, Higgins D (2007) Clustal W and Clustal X version 2.0. Bioinformatics 23:2947–2948
3. Ogurtsov A, Sunyaev S, Kondrashov A (2004) Indel-based evolutionary distance and mouse-human divergence. Genome Res 14:1610–1616
4. 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
5. Feng D, Doolittle R (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J Mol Evol 25:351–360
6. Löytynoja A, Goldman N (2008) Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. Science 320:1632–1635
7. Löytynoja A, Goldman N (2005) An algorithm for progressive multiple alignment of sequences with insertions. Proc Natl Acad Sci USA 102:10557–10562
8. Löytynoja A, Vilella A, Goldman N (2012) Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. Bioinformatics 28:1684–1691
9. Edgar R (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32:1792–1797
10. Liu K, Raghavan S, Nelesen S, Linder C, Warnow T (2009) Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science 324:1561–1564
11. Lee C, Grasso C, Sharlow MF (2002) Multiple sequence alignment using partial order graphs. Bioinformatics 18:452–464
12. Vialle RA, Tamuri AU, Goldman N (2018) Alignment modulates ancestral sequence reconstruction accuracy. Mol Biol Evol 35:1783–1797
13. Suchard M, Redelings B (2006) BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22:2047–2048
14. Li C, Medlar A, Löytynoja A (2016) Co-estimation of phylogeny-aware alignment and phylogenetic tree. bioRxiv, p. 077503
15. Blackburne BP, Whelan S (2012) Measuring the distance between multiple sequence alignments. Bioinformatics 28:495–502
16. Robinson DF, Foulds LR (1981) Comparison of phylogenetic trees. Math Biosci 53:131–147
17. Katoh K, Asimenos G, Toh H (2009) Multiple alignment of DNA sequences with MAFFT. In: Posada D (ed) Bioinformatics for DNA sequence analysis. Humana Press, Totowa, pp 39–64
18. Stamatakis A (2006) RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics 22:2688–2690
19. Price MN, Dehal PS, Arkin AP (2010) FastTree 2–approximately maximum-likelihood trees for large alignments. PLoS One 5:e9490
20. Fletcher W, Yang Z (2010) The effect of insertions, deletions, and alignment errors on the branch-site test of positive selection. Mol Biol Evol 27:2257–2267

21. Jordan G, Goldman N (2012) The effects of alignment error and alignment filtering on the sitewise detection of positive selection. Mol Biol Evol 29:1125–1139
22. Dessimoz C, Gil M (2010) Phylogenetic assessment of alignments reveals neglected tree signal in gaps. Genome Biol 11:R37
23. Landan G, Graur D (2007) Heads or tails: a simple reliability check for multiple sequence alignments. Mol Biol Evol 24:1380–1383
24. Novák A, Miklós I, Lyngsø R, Hein J (2008) StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. Bioinformatics 24:2403–2404
25. Veidenberg A, Medlar A, Löytynoja A (2016) Wasabi: an integrated platform for evolutionary sequence analysis and data visualization. Mol Biol Evol 33:1126–1130

Chapter 3

Fast and Accurate Multiple Sequence Alignment with MSAProbs-MPI

Jorge González-Domínguez

Abstract

Multiple sequence alignment (MSA) is a central step in many bioinformatics and computational biology analyses. Although there exist many methods to perform MSA, most of them fail when dealing with large datasets due to their high computational cost. MSAProbs-MPI is a publicly available tool (http://msaprobs.sourceforge.net) that provides highly accurate results in relatively short runtime thanks to exploiting the hardware resources of multicore clusters. In this chapter, I explain the statistical and biological concepts employed in MSAProbs-MPI to complete the alignments, as well as the high-performance computing techniques used to accelerate it. Moreover, I provide some hints about the configuration parameters that should be used to guarantee high-performance executions.

Key words Multiple sequence alignment, High-performance computing, MSAProbs-MPI, Parallel computing, Message passing interface, Multithreading

1 Introduction

The recent advances in next-generation sequencing (NGS) technologies have led to a scenario where large datasets that are used in a wide range of bioinformatics analyses. Multiple sequence alignment (MSA) is of central importance for many of them. For instance, it serves as the basis for the detection of homologous regions, for detecting motifs and conserved regions, for detecting structural building blocks, for constructing sequence profiles, and as an important prerequisite for the construction of phylogenetic trees. The use of exact methods to simultaneously align multiple sequences is impractical even for medium-size datasets. Instead, many methods based on different statistical concepts have been developed. There exist several of these methods to align multiple sequences. This chapter focuses on MSAProbs [1], which is based on pair hidden Markov models and partition function posterior probabilities. Several studies have confirmed the high accuracy of these tools [2–4] compared to other methods such as Clustal

Omega [4], MAFFT [5], MUSCLE [6], PRANK [7], BAli-Phy [8], NCBIBLAST [9], SSEARCH [10], FastSP [11], Probalign [12], Probcons [13], T-Coffee [14], Kalign [15], FSA [16], Dialign [17], and ClustalW [18].

Although MSAProbs is a statistical tool, its complexity is still high and its execution can be prohibitive for large datasets [4]. A parallel version of this tool called MSAProbs-MPI [19] was developed in order to overcome this problem. It is implemented using Message Passing Interface (MPI) [20] and OpenMP [21] in order to exploit the computational capabilities of multicore clusters and supercomputers. It provides the same accurate results as the original tool but in significantly lower time when several nodes are available.

In this chapter, I will provide and overview about how scientists can benefit from MSAProbs-MPI in order to obtain fast and accurate MSAs. I will start with a summary of the bioinformatics method under MSAProbs (*see* Subheading 2). Subheading 3 will describe the parallel implementation. I will finish the chapter with Subheading 4 where I will enumerate the options available to the users in MSAProbs and MSAProbs-MPI, as well as an execution example.

2 MSAProbs Method

MSAProbs receives as input one or several FASTA files with many sequences and provides the most relevant MSAs. It avoids checking the quality of all possible groups of sequences (which would be unfeasible for large datasets) thanks to applying a progressive alignment strategy with the following steps (more information in [1]):

1. Calculation of the posterior probability matrices for all pairs of sequences. Concretely, MSAProbs calculates for each pair two probability matrices. On the one hand, it uses a forward and backward algorithm [22] to calculate a matrix based on hidden Markov models (HMM). On the other hand, a second matrix with suboptimal alignments is generated through dynamic programming. The values of the posterior probability matrix for each sequence-pair is the root mean square of the value of that pair in the two previous matrices.
2. Calculation of a pairwise distance matrix that stores, for each pair, the optimal global alignment score. This score is calculated from the posterior matrix of the pair constructed in the previous step.
3. Construction of a guided tree from the pairwise distance matrix using the UPGMA clustering method [23].

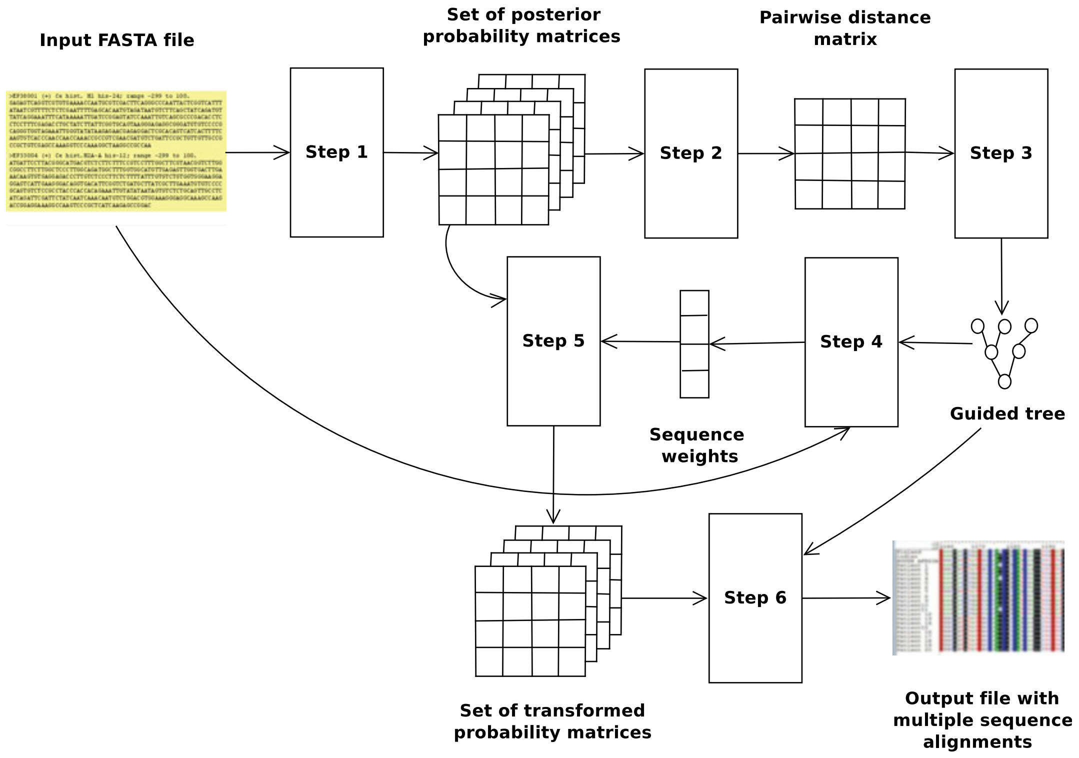

Input FASTA file
Set of posterior probability matrices
Pairwise distance matrix
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Guided tree
Sequence weights
Set of transformed probability matrices
Output file with multiple sequence alignments

Fig. 1 Workflow of MSAProbs

4. Calculation of the weight for each sequence according to the tree topology.
5. Transformation of all the posterior probability matrices by introducing a third sequence and using the weight of the three sequences.
6. Computation of a progressive alignment along the guide tree using the already transformed posterior probability matrices. A post-processing refinement can also be performed.

Figure 1 illustrates these steps. Remark that phases 1 and 5 are the most computationally demanding ones, with complexity of $O(N^2L^2)$ and $O(N^2L^3)$, respectively, being N the number or sequences and L the average sequence length.

3 Parallel Implementation in MSAProbs-MPI

The exploitation of high-performance computing (HPC) facilities is useful in order to accelerate the MSA procedure. In this section, I will provide a short overview of the parallel approach followed by MSAProbs [1] and MSAProbs-MPI [19].

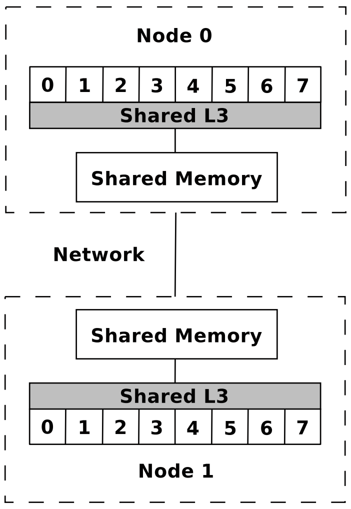


Fig. 2 Example of a multicore cluster with two nodes, each one with eight cores

3.1 Target Hardware

First, it is important to know the type of HPC facilities that can be used to accelerate MSAProbs and MSAProbs-MPI executions. Their target are multicore clusters, which contain several nodes, each one with several cores. An example of a small multicore cluster with two nodes can be seen in Fig. 2.

A cluster consists of several nodes, each one containing several cores (in the example of Fig. 2, eight cores per node). All the cores available in the system can work at the same time, either they are in the same or different nodes. All the cores within one node share the memory, i.e., they can directly access data that has been written by other core within the same node. However, each node has its own memory, and it is not directly accessible to the cores allocated in other node. It means that if one core requires data calculated by a core of a different node, this data must be sent through the interconnection network with a message. Therefore, in terms of memory, multicore clusters are classified as hybrid-memory systems, as they present both shared memory (within each node) and distributed memory (between nodes).

3.2 OpenMP

OpenMP [21] is a parallel programming interface based on a set of compiler directives. It follows a fork-join model, where the master/father thread creates a number of slaves/children threads (as shown in Fig. 3) that can perform different tasks in parallel and access the same shared memory. Task assignment to threads can be done statically (known at the beginning of the execution) or dynamically (new tasks are assigned once threads finish their previous tasks).

The main advantage of OpenMP is its simplicity, as the programmer only needs to find the most demanding portions of code and write the directives before them. These directives directly

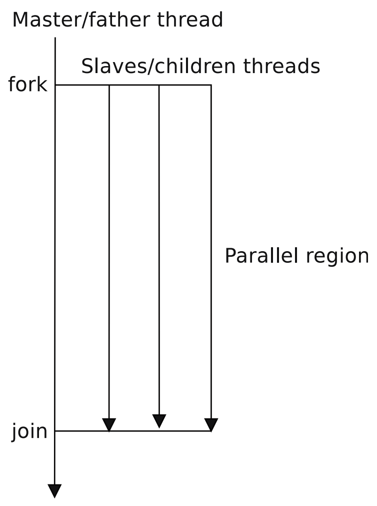


Fig. 3 Abstraction of the fork-join model followed by OpenMP

distribute the workload among the cores. Its main drawback is its low scalability, as it can only be used on systems with shared memory. Therefore, only cores within one node of a multicore cluster can collaborate in an OpenMP-based application.

3.3 Message Passing Interface (MPI)

MPI [20] is the most common programming model for distributed-memory systems. In fact, it is established as de facto standard for message-passing as it is based on the consensus of more than 40 organizations, including hardware vendors, researchers, and software developers. The MPI standard is currently in its third version. The success of MPI comes from its portability, efficiency, and flexibility to carry out message-passing. Note that MPI is only an interface definition that has been implemented by several developers for different architectures. Nowadays you can find a several implementations whose routines or functions can be directly called from C, C++, and Fortran codes.

A parallel MPI program consists of several processes with associated local memory. In a pure MPI program, each process is linked to one core, and each core has one piece of the memory available in the node assigned. In hybrid MPI and multithreaded programs, each process is usually mapped to one node, the local MPI memory is associated as the whole shared-memory of the node, and the MPI processes launch several associated threads (often the same number of threads as cores within the node). If the tasks of different processes are completely independent, they do not need to exchange information. Otherwise, data communication must be performed through the interconnection network. The traditional MPI communication style is two-sided, i.e., the source and destination processes must be synchronized through send and receive routines. MPI also provides collective routines for communication patterns that involve a group of processes. These collectives are usually very efficient as there exist optimized versions for specific architectures [24].

3.4 Parallel Approach

Both MSAProbs and MSAProbs-MPI have support for parallel computing. On the one hand, the original tool only includes multi-threading. Therefore, it can only exploit those resources that share memory (cores within one node of the cluster). On the other hand, MSAProbs-MPI extends this approach with MPI routines so that the workload can also be distributed among different nodes. As indicated in the previous section, each process is associated to a group of cores (for instance, all the cores within one node), and it launches several threads to map its tasks among the cores of the group. This hybrid approach has provided satisfactory results for applications related to different research fields such as medical imaging [25], genetics [26], genomics [27], or machine learning [28].

As mentioned at the end of Subheading 2, the calculation of the posterior probability matrices and their transformations (**steps 1** and **5**) are the most computationally demanding phases, and thus MSAProbs-MPI focused on applying parallel directives to them to reduce their runtime. All processes start reading the input matrix with efficient MPI I/O routines. Although all of them read the whole input matrix, the calculation of the posterior probability matrices and the pairwise distances is distributed among them. Concretely, all processes (and their associated threads) deal with the same number of sequence-pairs in order to balance the workload. At the end of the second stage (*see* Subheading 2), each process sends its portion of the distance matrix to Process 0, which gathers all the fragments. Process 0 sequentially constructs the guided tree (**step 3**), calculates the sequence weights (**step 4**), and broadcasts them to all processes with a collective operation. Parallelizing these third and fourth steps is not necessary as they are computationally negligible.

Again, the transformation of the posterior matrices is performed in parallel by several processes and threads. Concretely, a block-based approach with a ring communication pattern is used (more information can be found in [19]). Finally, the transformed matrices are sent to Process 0, which also completes the final alignment using several threads and writes the output into the file. Figure 4 shows again the MSAProbs-MPI workload but including information about which steps are parallelized.

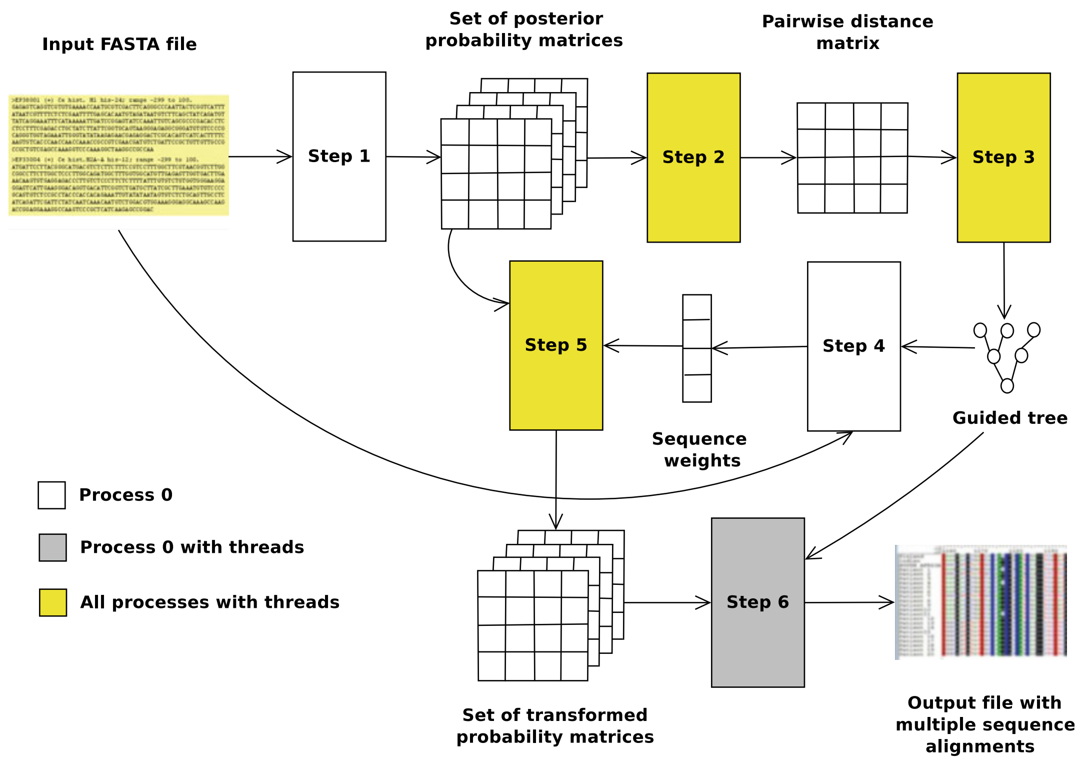

Input FASTA file
Step 1
Set of posterior probability matrices
Step 2
Pairwise distance matrix
Step 3
Guided tree
Step 4
Sequence weights
Step 5
Process 0
Process 0 with threads
All processes with threads
Set of transformed probability matrices
Step 6
Output file with multiple sequence alignments

Fig. 4 Workflow of MSAProbs-MPI. In white sequential steps. In gray steps with only OpenMP parallelization. In yellow steps with full MPI/OpenMP parallelization

4 Execution of MSAProbs-MPI

This section will help the future users of MSAProbs and MSAProbs-MPI to work with them. I focus on the second tool as it includes all the options of the original one and extends it with some additional ones. Those users that are able to work with MSAProbs-MPI have more than the necessary knowledge to work with MSAProbs.

4.1 Options for the Bioinformatics Method and the Parallel Implementation

Both MSAProbs and MSAProbs-MPI are configurable in order to adapt to different kinds of experiments. Here you have a list of the available parameters:

- -o, -outfile. String to specify the output file name (STDOUT by default).
- -num_threads. Integer with the number of threads per MPI process (1 by default). In the case of the original MSAProbs, it is the total number of threads.

- -b, -num_blocks. Integer with the number of blocks used for the transformation on each process. Less blocks obtain in general better performance but require more memory. This parameter is only available in MSAProbs-MPI.
- -clustalw. If specified, use CLUSTALW output format instead of FASTA format.
- -c, -consistency. Integer with the number of passes of consistency transformation (2 by default, 0 minimum, and 5 maximum).
- -ir, -iterative-refinement. Integer with the number of passes of iterative refinement (10 by default, 0 minimum, and 1000 maximum).
- -v, -verbose. If specified, report progress while alignment.
- -annot. String to specify an annotation file for the multiple alignment.
- -a, -alignment-order. If specified, print output sequences in alignment other rather than input order.
- -version. If specified print out version of the program.

4.2 Installation Instructions

To complete the installation of MSAProbs-MPI follow these steps:

1. Download the source code from http://msaprobs.sourceforge.net/.
2. Untar the archive and move into the MSAProbs-MPI directory.
3. Update the file Makefile of the root directory in order to indicate the correct path and libraries for the MPI compiler installed in your system.
4. Type make to build MSAProbs-MPI.

Ask administrator of your computer system in case you have problems with the MPI and/or OpenMP installation.

4.3 Execution Example

For instance, the following command prints in the file *outAlign.fasta* the multiple sequence alignment of *dataset1.fasta* on 8 cores using 2 MPI processes with 4 OpenMP threads each, 3 consistency transformation passes, and 100 iterations for refinement. This *dataset1.fasta* is a test example available with the source code at http://msaprobs.sourceforge.net/; an mpirun is the command used to run MPI in the system in many implementations (for instance, OpenMPI).

mpirun -n 2 ./msaprobs-mpi dataset1.fasta -num threads 4 -c 3 -it 100 -o outAlign.fasta.

References

1. Liu Y, Schmidt B, Maskell DL (2010) MSA-Probs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. Bioinformatics 26 (16):1958–1964
2. Katoh K, Standley DM (2016) A simple method to control over-alignment in the MAFFT multiple sequence alignment program. Bioinformatics 32(13):1933–1942
3. Rivas E, Eddy SR (2015) Parameterizing sequence alignment with an explicit evolutionary model. BMC Bioinformatics 16(1):406
4. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W et al (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7(1):539
5. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30:772–780
6. Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32 (5):1792–1797
7. Löytynoja A, Goldman N (2008) Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. Science 320(5883):1632–1635
8. Redelings B (2014) Erasing errors due to alignment ambiguity when estimating positive selection. Mol Biol Evol 31(8):1979–1993
9. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K et al (2009) BLAST +: Architecture and applications. BMC Bioinformatics 10:421
10. Pearson WR (2000) Flexible sequence similarity searching with the FASTA3 program package. Methods Mol Biol 132:185–219
11. Mirarab S, Warnow T (2011) FastSP: Linear time calculation of alignment accuracy. Bioinformatics 27:3250–3258
12. Roshan U, Livesay DR (2006) Probalign: multiple sequence alignment using partition function posterior probabilities. Bioinformatics 22 (22):2715–2721
13. Do CB, Mahabhashyam MSP, Brudno M, Batzoglou S (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. Genome Res 15:330–340
14. Notredame C, Higgins DG, Heringa J (2000) T-Coffee: A novel method for fast and accurate multiple sequence alignment. J Mol Biol 302 (1):205–217
15. Lassmann T, Sonnhammer ELL (2005) Kalign - an accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics 6:298
16. Bradley RK, Roberts A, Smoot M, Juvekar S, Do J, Dewey C, Holmes I, Pachter L (2009) Fast statistical alignment. PLoS Comput Biol 5:e1000392
17. Morgenstern B, Frech K, Dress A, Werner T (1998) DIALIGN: finding local similarities by multiple sequence alignment. Bioinformatics 14:290–294
18. Larkin MA, Blackshields G, Brown NP, Chenna R, McGettigan PA, McWilliam H, Valentin F, Wallace IM, Wilm A, Lopez R, Thompson JD, Gibson TJ, Higgins DG (2007) Clustal W and Clustal X version 2.0. Bioinformatics 23:2947–2948
19. González-Domínguez J, Liu Y, Touriño J, Schmidt B (2016) MSAProbs-MPI: parallel multiple sequence aligner for distributed-memory systems. Bioinformatics 32 (24):3826–3828
20. MPI Forum (2012) MPI: a message-passing interface standard version 3.0. Technical Report, University of Tennessee, Knoxville
21. Dagum L, Menon R (1998) OpenMP: an industry-standard API for shared-memory programming. Comput Sci Eng 1:46–55
22. Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
23. Sneath PH, Sokal RR (1973) Numerical taxonomy. The principles and practice of numerical classification
24. Tu B, Fan J, Zhan J, Zhao X (2012) Performance analysis and optimization of MPI collective operations on multi-core clusters. J Supercomput 60(1):141–162
25. González-Domínguez J, Remeseiro B, Martín MJ (2017) Parallel definition of tear film maps on distributed-memory clusters for the support of dry eye diagnosis. Comput Methods Prog Biomed 139:51–60
26. González-Domínguez J, Martín MJ (2017) MPIGeneNet: parallel calculation of gene co-expression networks on multicore clusters. IEEE/ACM Trans Comput Biol Bioinform 15 (5):1732–1737
27. Ponte-Fernández C, González-Domínguez J, Martín MJ (2019) Fast search of third-order epistatic interactions on CPU and GPU clusters. The International Journal of High Performance Computing Applications, Online
28. González-Domínguez J, Bolón-Canedo V, Freire B, Touriño J (2019) Parallel feature selection for distributed-memory clusters. Inf Sci 496:399–409

Part II

Tools for Specific MSA Problems

Chapter 4

Aligning Protein-Coding Nucleotide Sequences with MACSE

Vincent Ranwez, Nathalie Chantret, and Frédéric Delsuc

Abstract

Most genomic and evolutionary comparative analyses rely on accurate multiple sequence alignments. With their underlying codon structure, protein-coding nucleotide sequences pose a specific challenge for multiple sequence alignment. Multiple Alignment of Coding Sequences (MACSE) is a multiple sequence alignment program that provided the first automatic solution for aligning protein-coding gene datasets containing both functional and nonfunctional sequences (pseudogenes). Through its unique features, reliable codon alignments can be built in the presence of frameshifts and stop codons suitable for subsequent analysis of selection based on the ratio of nonsynonymous to synonymous substitutions. Here we offer a practical overview and guidelines on the use of MACSE v2. This major update of the initial algorithm now comes with a graphical interface providing user-friendly access to different subprograms to handle multiple alignments of protein-coding sequences. We also present new pipelines based on MACSE v2 subprograms to handle large datasets and distributed as Singularity containers. MACSE and associated pipelines are available at: https://bioweb.supagro.inra.fr/macse/.

Key words Multiple sequence alignment, Molecular evolution, Phylogenomics, Pseudogenes, Metabarcoding, Bioinformatics pipelines

1 Introduction

Multiple sequence alignment (MSA) is a crucial step in many evolutionary analyses. Nonetheless, the most commonly used alignment tools overlook the underlying codon structure of protein-coding nucleotide sequences. Accounting for this structure is useful for improving the proposed alignment, but it is also a prerequisite for some downstream analyses such as selection pressure analysis based on the nonsynonymous to synonymous substitution ratio (dN/dS).

MACSE [1] was specifically designed to align protein-coding nucleotide (NT) sequences with respect to their amino acid (AA) translation while allowing NT sequences to contain multiple frameshifts and/or stop codons (*see* Fig. 1). MACSE thus provided the first automatic solution for aligning protein-coding gene datasets containing nonfunctional sequences (pseudogenes) without

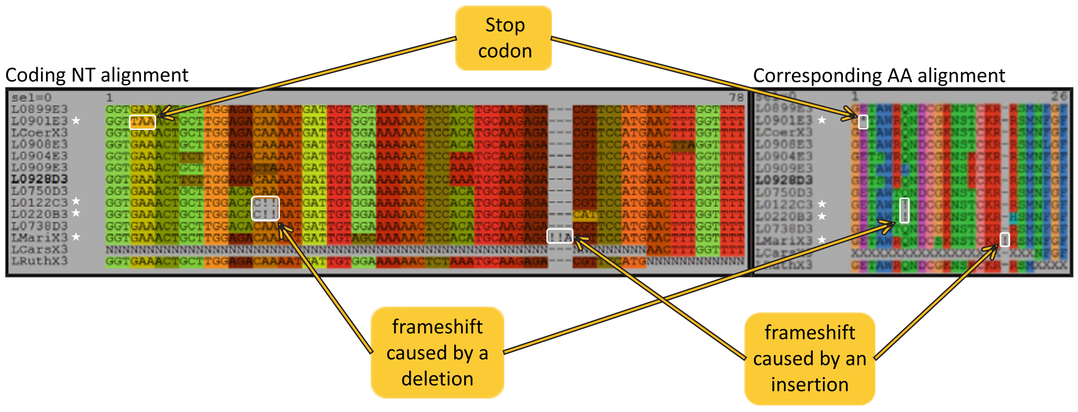

Stop codon
Coding NT alignment
Corresponding AA alignment
frameshift caused by a deletion
frameshift caused by an insertion

Fig. 1 MACSE alignment of a set of nucleotide sequences containing functional genes as well as pseudogenes (marked by a white star). The nucleotide alignment (NT) and its amino acid (AA) translation are edited with SeaView (‘codon-colors’ option for the NT alignment). Frameshifts caused by deletions and insertions are represented by the ‘!’ character. A white frame highlights Frameshifts and stop codons

disrupting the underlying codon structure. It has also proved useful in detecting undocumented frameshifts in public database sequences and in aligning next-generation sequencing reads/contigs against reference coding sequences [2], especially for metabarcoding analysis [3].

The first MACSE release contained a single program that took coding nucleotide sequences as input and aligned them with respect to their codon structures [1]. This early command line version included multiple options that allowed end-users to fine-tune the alignment options, but its use could be tedious. In order to streamline the program application, we built several companion tools that exploit the core MACSE algorithm to tackle related problems [4]. The resulting MACSE v2 toolkit was hence much more powerful as it provided the building blocks to construct powerful alignment pipelines. However, the number of available subprograms and options featured in this version was problematic for occasional users. We finally proposed a Graphical User Interface (GUI) to improve the end-user experience. This GUI is useful for new users who can test MACSE on a few datasets without first having to deal with the command line option complexity. Moreover, the GUI displays the command line corresponding to selected options, thus streamlining the transition from the GUI to the command line version.

When aligning protein-coding nucleotide sequences, it is often necessary to chain several steps such as sequence prefiltering (e.g., to remove unwanted UTR fragments) and then producing and filtering the nucleotide alignment based on its amino acid translation. We have successfully used MACSE to design effective pipelines for various tasks, such as aligning thousands of orthologous sequence datasets from the OrthoMaM database [5] or correcting tens of thousands of barcoding reads [6]. In this chapter, we introduce the specificity and key functionalities of MACSE v2. We outline some standard use cases to illustrate how MACSE subprograms can be chained to produce high-quality protein-coding sequence alignments in various contexts. All examples mentioned in this chapter can be downloaded from the MACSE website https://bioweb.supagro.inra.fr/macse/. The two main pipelines discussed here are also available as Singularity containers [7] for easy installation and use on high-performance computing clusters.

2 MACSE Basic Usage and Possible Troubleshooting

2.1 Getting Started

MACSE is written in JAVA and hence runs in a straightforward way on any computer that has a Java Runtime Environment (JRE) release installed. If needed, JRE is available for free download on the Java website (www.java.com). The most recent MACSE release is then available for download on the MACSE website (https://bioweb.supagro.inra.fr/macse). This website also contains detailed documentation with several examples for each subprogram, as well as detailed explanations of possible applications. Each MACSE release is a single jar file. The latest 2019 release is macse_v2.03.jar. It can be launched by typing the following command:

java -jar macse_v2.03.jar

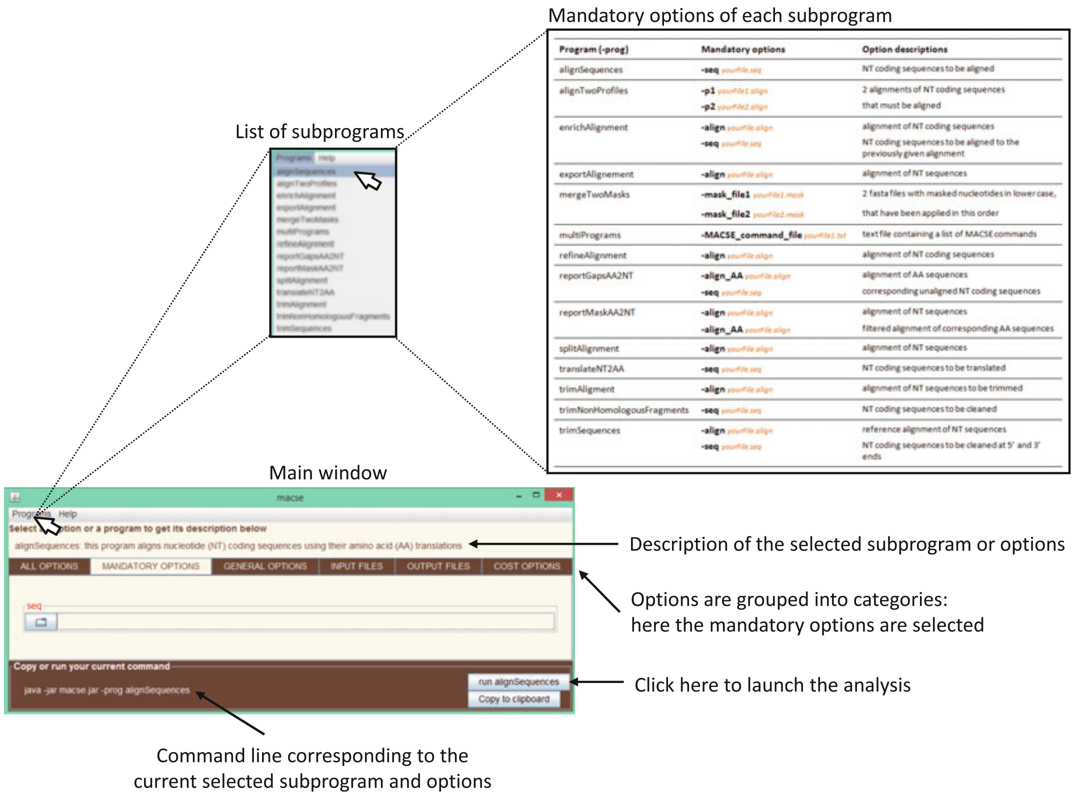
⇨ Launches the GUI version of MACSE (*see* Fig.2).

MACSE may also be launched by double clicking on the macse_v2.03.jar file. In both cases this will launch the graphic user interface of MACSE. Anything typed after macse_v2.03.jar will be considered as options passed to MACSE, whereas anything typed before will be considered as Java virtual machine options. The command line version and GUI versions of MACSE may be run via the same MACSE jar file. In the absence of any option, the GUI version is launched, whereas the command line version is launched as soon as at least one option is submitted to MACSE. As MACSE is a set of subprograms, the "-prog" option allows users to specify the subprogram to be executed. This is a mandatory option, but if the user does not know the subprogram names, any name may be submitted, and a help message with a list of possible subprograms will be displayed:

java -jar macse_v2.03.jar -prog wrongProgram

⇨ Launches the command line version of MACSE, and print a help message listing all valid subprograms with a one-line description of each of them.

Once the name of the subprogram of interest has been selected, e.g., alignSequences, a brief help message for this subprogram can



Program (-prog)	Mandatory options	Option descriptions
alignSequences	**-seq** yourfile.seq	NT coding sequences to be aligned
alignTwoProfiles	**-p1** yourfile1.align **-p2** yourfile2.align	2 alignments of NT coding sequences that must be aligned
enrichAlignment	**-align** yourfile.align **-seq** yourfile.seq	alignment of NT coding sequences NT coding sequences to be aligned to the previously given alignment
exportAlignement	**-align** yourFile.align	alignment of NT sequences
mergeTwoMasks	**-mask_file1** yourFile1.mask **-mask_file2** yourFile2.mask	2 fasta files with masked nucleotides in lower case, that have been applied in this order
multiPrograms	**-MACSE_command_file** yourfile1.txt	text file containing a list of MACSE commands
refineAlignment	**-align** yourFile.align	alignment of NT coding sequences
reportGapsAA2NT	**-align_AA** yourFile.align **-seq** yourFile.seq	alignment of AA sequences corresponding unaligned NT coding sequences
reportMaskAA2NT	**-align** yourFile.align **-align_AA** yourFile.align	alignment of NT sequences filtered alignment of corresponding AA sequences
splitAlignment	**-align** yourFile.align	alignment of NT sequences
translateNT2AA	**-seq** yourFile.seq	NT coding sequences to be translated
trimAligment	**-align** yourFile.align	alignment of NT sequences to be trimmed
trimNonHomologousFragments	**-seq** yourFile.seq	NT coding sequences to be cleaned
trimSequences	**-align** yourFile.align **-seq** yourFile.seq	reference alignment of NT sequences NT coding sequences to be cleaned at 5' and 3' ends

Select an option or a program to get its description below

alignSequences: this program aligns nucleotide (NT) coding sequences using their amino acid (AA) translations

```
java -jar macse.jar -prog alignSequences
```

Fig. 2 Presentation of the MACSE Graphical User Interface showing the different parts of the main window: the "program" menu allowing users to choose the subprograms accompanied by a table listing all of them (with their mandatory options and the required files) and the location where each element can be found (where a brief description of the selected subprogram or option can be found, the different menus, the command line, etc.)

be displayed by invoking it without further options, and a description of what this subprogram is useful for and a list of mandatory options will be printed:

java -jar macse_v2.03.jar -prog alignSequences

⇨ Prints a basic help message of the alignSequences subprogram focusing on its mandatory options.

The "-help" option provides more detailed information and the complete list of options:

java -jar macse_v2.03.jar -prog alignSequences -help

⇨ Prints a detailed help message of the alignSequences subprogram presenting all available options.

Documentation may also be accessed when using GUI (*see* Fig. 2). Once the subprogram of interest is selected via the "Programs" menu, a brief description of this subprogram appears at the top of the GUI. Options are grouped into categories: mandatory options, output file names, alignment parameters, etc. Once an option field is selected by clicking on it, the related documentation is displayed at the top of the GUI. The command line corresponding to the graphically selected options appears at the bottom of the GUI. Copying this command line before running MACSE via the GUI ensures the traceability of the analysis while also enabling the user to easily run the same analysis via the command line without having to manually type the command line.

Hereafter we shorten the command line by omitting the MACSE release version. Note that this can also be done by renaming the downloaded jar file by using a symbolic link, by defining an environment variable on the system, or through any other technical solution that suits the user. For enhanced readability, we also extend the command to several lines, with one option per line, and indicate the option name in bold font. It follows that a command such as:

java -jar macse_v2.03.jar -prog alignSequences -help

will hence be written in the rest of this chapter as:

```
java -jar macse.jar -prog alignSequences
                    -help
```

2.2 Obtaining Suitable Input Sequences

The most frequent pitfall encountered by new MACSE users arises when the user provides an input sequence file containing fragments of nonprotein-coding nucleotide sequences. In case of unexpected MACSE behavior, the first thing to check is that the input sequence file contains nucleotide sequences in a valid fasta format. To do so, users may try to open it with a sequence/alignment viewer such as SeaView [8] or AliView [9], which are very convenient to visualize sequences and alignments produced by MACSE. These viewers accept the '!' character in both nucleotide and amino acid sequences, and it is also possible to visually highlight the codon structure of the aligned nucleotide sequences.

A second aspect to verify is that the sequences are all in forward direction. This could be harder to check depending on how the sequences have been obtained, but MACSE will not be able to correctly align sequences in reverse orientation. A solution could be to blast them against public protein databases using blastx. All sequences for which the best hit occurs with a negative reading frame should probably be reverse translated. Alternatively, MAFFT [10] has convenient functionalities (--adjustdirection or --adjustdirectionaccurately) that can reorient nucleotide sequences in a multiple sequence alignment.

The last point is to ensure that the input sequences do not contain nonprotein-coding fragments. Typically, nonprotein-coding fragments in CDS are found when UTRs (or introns) are not trimmed out. This often occurs when dealing with de novo assembled contigs. Contigs should have their nonprotein-coding parts removed before alignment with MACSE. This could be done using dedicated annotation tools such as prot4EST [11], UTRme [12], or other similar tools. Alternatively, the MACSE trimNonHomologousFragments subprogram may be used. This subprogram will not focus specifically on noncoding regions, but it will mask any long fragment that is nonhomologous (at the amino acid level) to other sequences.

The trimNonHomologousFragments subprogram was initially developed to filter long insertions that may be caused, for instance, by annotation errors such as undetected introns or UTRs. Having to handle long insertions in one or a few sequences could drastically slow down the alignment process. Alignment of these nonhomologous regions is mostly useless, as they would probably be removed by any alignment filtering tools in subsequent analyses.

The trimNonHomologousFragments subprogram mainly aims at removing long nonhomologous fragments but keeps smaller ones to limit the risk of removing fragments that are actually homologous. Several options are provided to adjust the stringency of this prefiltering step, but we advise against being too strict at this early stage of the analysis. At this stage, a sequence that has been trimmed along almost its entire length is likely not at all homologous to other sequences, so it might be better to remove it completely. For a sequence to be kept in the output fasta file, the percentage of this sequence that should remain after homology prefiltering can be adjusted (-min_homology_to_keep_seq). Full details of this prefiltering process can be output in a fasta file (-out_mask_detail) in which the original sequences are written using a mix of upper case (for preserved nucleotides) and lower case (for removed nucleotides) letters. In any case, the trimNonHomologousFragment subprogram outputs a CSV file summarizing the impact of this prefiltering process on each sequence. This file contains the number of nucleotides (including, or not, noninformative "N" nucleotides) that have been removed from the whole sequence and from its extremities. Note that the name of this output file can be specified (-out_trim_info option):

```
java -jar macse.jar -prog trimNonHomologousFragments
                    -seq ENSG00000125812_GZF1_raw.fasta
                    -out_trim_info output_stats.csv
                    -min_homology_to_keep_seq 0.6
```

⇨ Prefilters long nonhomologous sequence fragments; if more than 60% of a sequence is filtered then this sequence is entirely removed.

2.3 Most Common Usages

The alignSequences subprogram is the core feature of the MACSE v2 toolkit. Its single mandatory option is a fasta file containing the coding nucleotide sequences to align. These nucleotide sequences need to be in forward direction, as alignSequences ignores their reverse complements, and they should be protein-coding sequences all along. Indeed, as alignSequences relies on sequence protein translations to align sequences, if there are any UTR or intron fragments, alignSequences would waste a lot of time producing meaningless alignments.

To align CDSs of the Pg3 gene in the *Medicago* genus [13] stored in the fasta file named Pg3_Medicago.fasta, the simplest command line is:

```
java -jar macse.jar -prog alignSequences
                    -seq Pg3_Medicago.fasta
```

⇨ Aligns sequences contained in the Pg3_Medicago.fasta file with default parameters (*see* Fig. 1).

The alignSequences subprogram, like most other MACSE subprograms, generates two fasta files, one containing the aligned protein-coding nucleotide sequences as codons and another containing the corresponding amino acid alignment. By default, the names of these files are based on the input file name, but the desired output file names can be specified using the "-out_NT" and "-out_AA" options.

Since MACSE relies on amino acid translation, it lets you specify the genetic code adapted to your protein-coding sequences. The NCBI has assigned a unique number to each genetic code, which is convenient to easily specify which code should be used. By default, MACSE uses "the standard code," but a different default genetic code may be specified for a dataset using the "-gc_def" option. For instance, the invertebrate mitochondrial code is the fifth on the NCBI list. The command line below is hence adapted to align mitochondrial COX1 sequences of grasshoppers:

```
java -jar macse.jar -prog alignSequences
                    -seq grasshoppers_COX1.fasta
                    -gc_def 5
```

⇨ Aligns invertebrate mitochondrial sequences with the specified genetic code "5".

If the dataset contains sequences that use different genetic codes, they will have to be specified in a separated text file ("-gc_file" option) containing, on each line, the name of a sequence and the number of the corresponding genetic code. Any sequence absent from this file will be translated using either the genetic code specified by the -gc_def option or, in the absence of this option, the standard genetic code. For example, to align metazoan mitochondrial COX1 sequences from different phyla [14], the following command may be used to specify the five different genetic codes with the -gc_file option:

```
java -jar macse.jar -prog alignSequences
                    -seq Singh2009_cox1.fasta
                    -gc_file Singh2009_cox1_gc_file.txt
                    -out_NT Singh2009_cox1_NT.fasta
                    -out_AA Singh2009_cox1_AA.fasta
```

⇨ Aligns metazoan mitochondrial sequences with their corresponding genetic codes (*see* Fig. 3).

The translateNT2AA subprogram could also be used to simply translate protein-coding sequences using either the default standard genetic code if not specified or the genetic code specified using the -gc_def and -gc_file options:

```
java -jar macse.jar -prog translateNT2AA
                    -seq Singh2009_cox1.fasta
                    -gc_file Singh2009_cox1_gc_file.txt
```

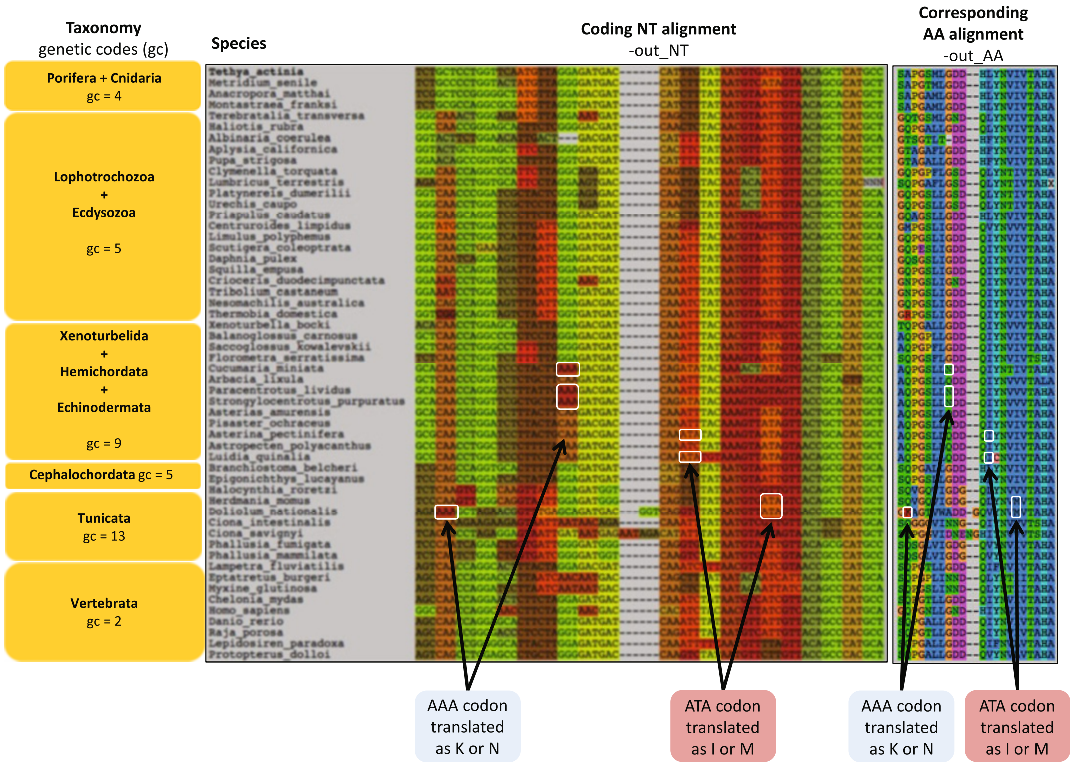

Taxonomy
genetic codes (gc)
Species
Coding NT alignment
-out_NT
Corresponding AA alignment
-out_AA
Porifera + Cnidaria gc = 4
Lophotrochozoa + Ecdysozoa gc = 5
Xenoturbelida + Hemichordata + Echinodermata gc = 9
Cephalochordata gc = 5
Tunicata gc = 13
Vertebrata gc = 2
AAA codon translated as K or N
ATA codon translated as I or M
AAA codon translated as K or N
ATA codon translated as I or M

Fig. 3 MACSE alignment of 54 metazoan mitochondrial COX1 sequences from Singh et al. [14] using five different mitochondrial genetic codes corresponding to the different taxonomic groups. The nucleotide alignment (NT) and its amino acid (AA) translation are edited with SeaView ('codon-colors' option for the NT alignment)

⇨ Translates metazoan mitochondrial sequences with their corresponding genetic codes.

The key options described so far are present in most MACSE subprograms.

Another set of options concerns the costs used to compare alternative alignments and select the best one. Like most alignment software, MACSE lets users tune the ratio between gap extension cost and gap opening cost. Increasing the gap opening cost (or decreasing the gap extension cost) will tend to favor alignments where gaps are grouped in long stretches. MACSE also allows adjustment of the relative cost of gaps appearing at the sequence extremities (terminal gaps) as opposed to those appearing inside the sequences (internal gaps). By default, external gaps are less penalized as they often reflect the fact that a sequence was partially sequenced rather than that a nucleotide insertion/deletion has occurred. Similarly, one or two missing nucleotides at the sequence extremities lead to incomplete codons (hence technically frameshifts), but such external frameshifts should not be as penalized as those occurring in the middle of a sequence (internal frameshifts). When a dataset contains a mix of genes and pseudogenes or of high-quality sequences (e.g., a CDS from the Swiss-Prot database) and low-quality sequences (e.g., de novo assembled contigs), it is also relevant to assign different penalties for the frameshifts and stop codons appearing in such different types of sequence. To deal with such cases, MACSE allows users to define two sets of sequences by providing two fasta files as input instead of a single one. The most reliable sequences are in the file provided by the "-seq" options, whereas the least reliable ones are in the file provided by the "-seq_lr" option. As it allows stop codons and frameshifts and allows users to assign them different penalty costs based on the sequence in which they appear and on their position within this sequence, MACSE features many more cost-related options than usual alignment software. These different cost options are summarized in Tables 1 and 2.

As frameshifts and stop codons are much less unexpected in pseudogenes than in nucleotide sequences coding for a functional protein, users may opt to decrease the cost of such events. For instance, the following parameters and options may be used to

Table 1
MACSE options to adjust stop codon and frameshift costs in sequences

	Internal Frameshift	Stop	Terminal Frameshift	Stop
Reliable sequences	-fs	-stop	-fs_term	--
Less reliable sequences	-fs_lr	-stop_lr	-fs_lr_term	--

Table 2
MACSE options to adjust gap costs

	Internal gap		Terminal gap	
Sequence\event	Opening	Extension	Opening	Extension
Any sequences	-gap_op	-gap_ext	-gap_op_term	gap_ext_term

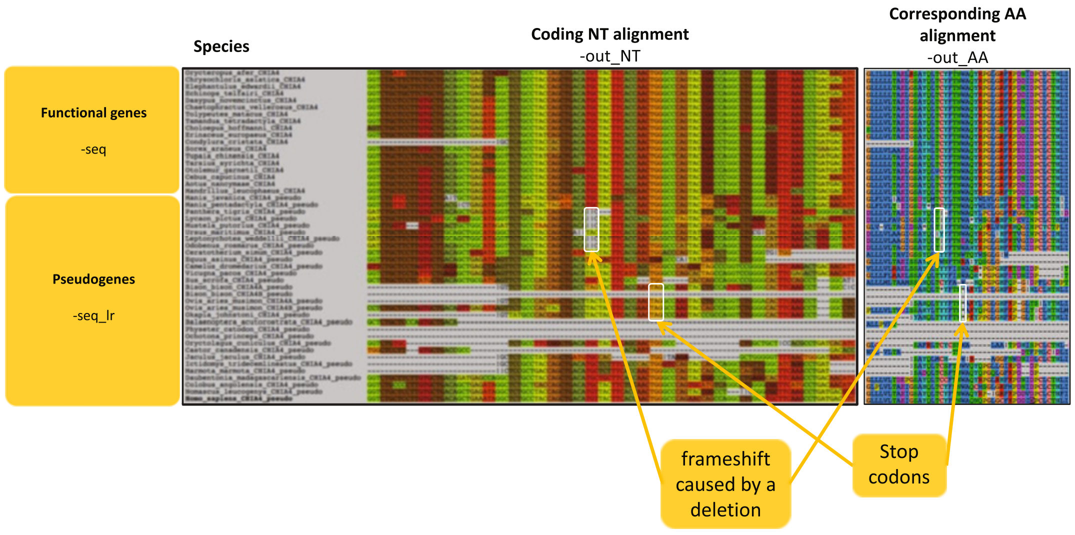


Fig. 4 MACSE alignment of 48 mammalian CHIA4 nucleotide sequences from Emerling et al. [15] containing 18 functional genes and 30 pseudogenes (_pseudo). The nucleotide alignment and its AA translation are edited with SEAVIEW ("codon-colors" option for the NT alignment). Frameshifts caused by deletions are represented by the "!" character. A white frame highlights Frameshifts and stop codons

align both functional and pseudogenized sequences from the mammalian CHIA4 gene [15]:

```
java -jar macse.jar -prog alignSequences
        -seq Emerling2018_CHIA4_functional.fasta
        -seq_lr Emerling2018_CHIA4_pseudo.fasta
        -fs_lr 10
        -stop_lr 10
        -out_NT Emerling2018_CHIA4_NT.fasta
        -out_AA Emerling2018_CHIA4_AA.fasta
```

⇨ Aligns a mix of functional CDS and pseudogenes (*see* Fig. 4).

The default parameters work fine for most cases, but in the MACSE online documentation, we provide some guidelines to help adjust parameter costs for some specific types of sequence datasets. Note that the default values for each parameter appear in the GUI.

3 MACSE-Based Pipelines Suitable for Datasets of Various Sizes

3.1 Pipelines Based on MACSE as Singularity Containers

We designed MACSE V2 as a toolkit dedicated to multiple alignments of protein-coding sequences that can be leveraged via both the command line and a Graphical User Interface (GUI). We used this toolkit to develop some convenient pipelines as described in this chapter. We share these pipelines as Singularity containers [7] since they also depend on a few other tools and some environment setups. A Singularity container contains everything needed to execute a specific task. The developer building the container has to handle dependencies and the environment configuration so that end-users will not need to worry about this. To run a Singularity container named "container.sif," that is, in your current directory, just type the following command in your Linux terminal:

singularity run ./container.sif

3.2 Basic Pipelines and Batch Facilities

Using the command line version of MACSE, it is quite easy for bioinformaticians to build an analysis pipeline chaining multiple MACSE subprograms to conduct tailored-made analyses on several input datasets. Scripting language or, even better, workflow managers are tools of choice for such tasks, but not everyone masters such tools. The "multiPrograms" subprogram of MACSE allows basic scripting for nonbioinformaticians. Its main option (-MACSE_command_file) allows specifying the file containing a list of MACSE commands that will be run sequentially. Each line of this command file must contain a single MACSE command starting by "-prog" (i.e., omitting "java -jar macse.jar"). The "@" character can be used before each file path to point towards the directory containing the command file itself (useful if the command file is not in the current directory). To prepare this command file, the end-user can apply the GUI on a single example to generate the required command line, copy this command line (using copy/paste or the "copy to clipboard" button) multiple times into a text file, and then replace the initial dataset name by a different one on each line. The basic usage of this subprogram is:

*java -jar macse.jar **-prog** multiPrograms*
*-**MACSE_command_file** align_multi.macse*

⇨ Launches all MACSE commands stored in the align_multi.macse file; for instance, to align sequences from three loci this file contains three lines:

```
-prog alignSequences -seq LOC_19470.fasta
-prog alignSequences -seq LOC_48720.fasta
-prog alignSequences -seq LOC_72220.fasta
```

When dealing with amino acid translation of nucleotide coding sequences, it is necessary to handle a larger alphabet (20 amino acids versus only 4 possible nucleotides), but then the sequences are three times shorter. However, because MACSE aligns protein-coding nucleotide sequences while accounting for their amino acid translations in the three possible reading frames, it needs to cope with longer nucleotide sequences and a larger amino acid alphabet. Moreover, most algorithmic optimizations of amino acid sequence alignment rely on the fact that their amino acid sequences are invariable, and gaps can be inserted only between amino acids. This means that amino acids never change throughout the alignment process. This is not the case with MACSE because frameshifts can potentially be introduced anywhere in a sequence, at any step of the alignment process. Amino acids of a given nucleotide sequence could therefore vary during the alignment process depending on the reading frames used at a given stage to translate the sequence. Optimizations generally used in alignment software are thus harder to incorporate into MACSE because the amino acid sequences may vary along the alignment process and different reading frames can be used to translate a single sequence. This specificity is a powerful feature of MACSE, but it increases the memory requirements and computation times. Thus, for datasets containing numerous long sequences, using the core alignSequences subprogram of MACSE with default options may not be feasible. In such cases, the alignSequences subprogram could be run to obtain a draft alignment that will hopefully unravel most frameshifts. Different strategies are presented in the following section to get the most of MACSE when dealing with datasets of various sizes.

MACSE is run through the Java virtual machine, so for relatively large datasets the memory that Java is allowed to use will have to be increased via the "-Xmx" option. This is not a MACSE option per se, but it is definitely essential:

java -jar ***–Xmx 600m*** *macse.jar* ***-prog*** *alignSequences*

⇨Aligns larger datasets by allocating more memory to Java using the Xmx option.

3.3 Aligning Dozens of Sequences

If the dataset is not too large, MACSE can be used to perform the whole alignment itself. We advise using this strategy, when possible, to get the most accurate frameshift placements. The command line for such an analysis could be as simple as launching MACSE with default options and allocating some extra memory for the Java virtual machine:

*java -jar **–Xmx 600m** macse.jar **-prog** alignSequences*
__-seq__ Pg3_Medicago.fasta

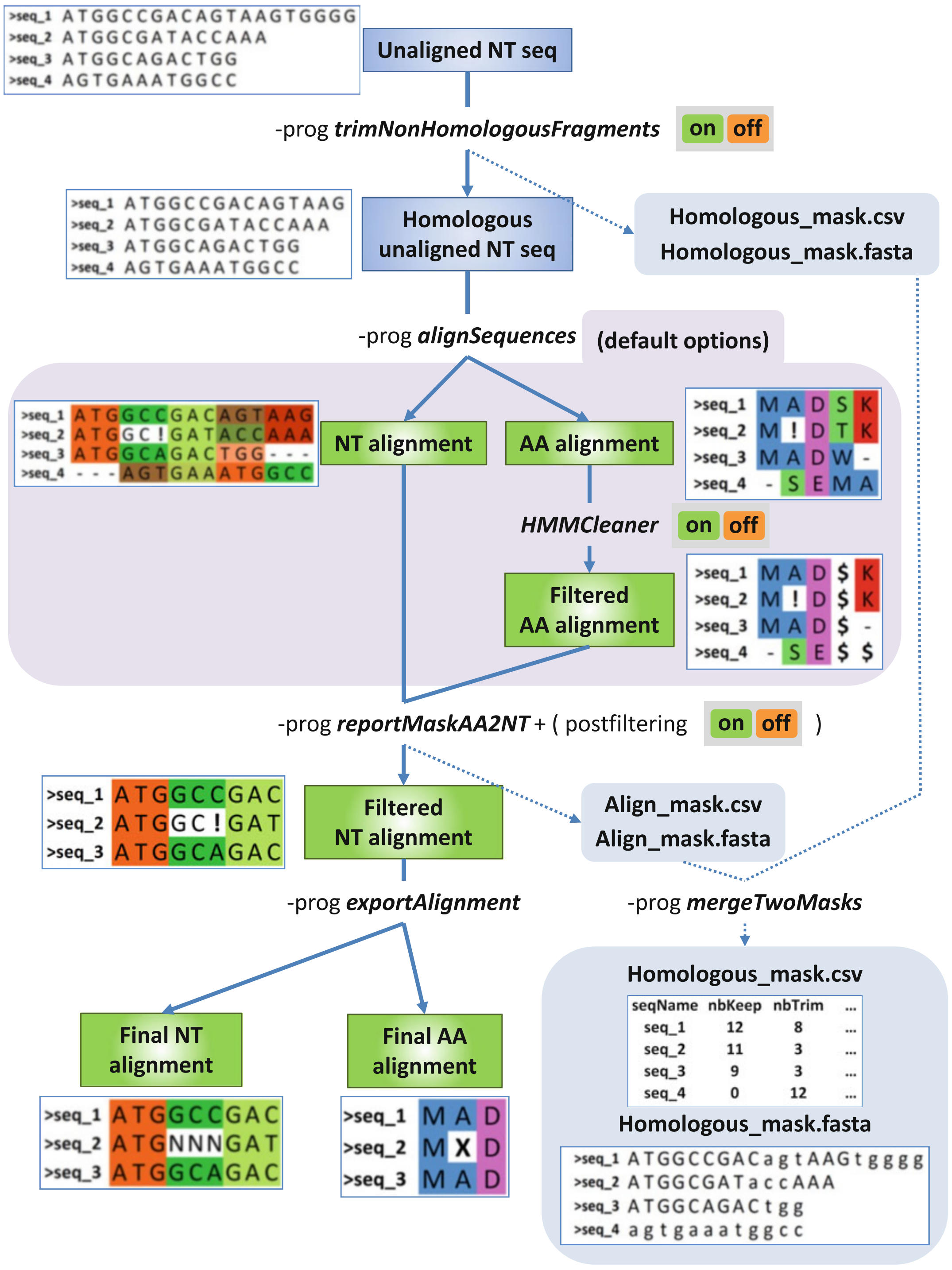
⇨ The most simple MACSE use case.

However, in most cases, it could be worth prefiltering possible UTRs or other long nonhomologous fragments contained in the sequences using the trimNonHomologousFragment MACSE subprogram. Some analyses, e.g., dN/dS estimation, are highly sensitive to alignment errors, which are favored by the presence of even short nonhomologous fragments. For such analyses, we strongly advise [16] also using HMMCleaner [17] to post-filter less reliable parts of your amino acid alignment and report this masking/filtering at the nucleotide level. The filtered alignment obtained with HMMCleaner may contain some isolated codons, surrounded only by gaps or masked codons, as well as sequences with very few remaining codons. It would make sense to remove such sequences and filter isolated codons. The reportMaskAA2NT subprogram of MACSE may be used to report the filtering performed by HMMCleaner at the nucleotide level and to perform some post-processing filtering of such isolated codons and patchy sequences. By using MACSE subprograms for these various filtering steps, the traceability of the filtering process is achieved by keeping track of every single nucleotide that has been masked. Finally, it could be convenient to be able to observe frameshifts and stop codons in the final alignment, but their presence might be problematic for downstream analyses. The alignments obtained with MACSE may be post-processed to replace stop codon and frameshift symbols by more standard ones using the exportAlignment subprogram of MACSE. Producing a reliable alignment of a dataset may hence require chaining several steps using HMMCleaner together with multiple MACSE subprograms. We provide a pipeline to automatize this process, while letting end-users turn on or off the various filtering steps. The script, written in Bash, is encapsulated in a Singularity container.

We called this pipeline MACSE_ALFIX (*see* Fig. 5), since it is mostly based on MACSE and chains the ALigning, Filtering, and eXporting steps. The script produces several output files that are stored in a single directory and named using a common prefix. The three mandatory options of this script are therefore the input file name, the output directory name, and the prefix of the output file names.

singularity run ./MACSE_ALFIX_v01.sif
__--in_seq_file__ LOC_48720.fasta
__--out_dir__ RES_LOC_48720
__--out_file_prefix__ LOC_48720

⇨ The most simple use case of the MACSE_ALFIX pipeline.



seqName	nbKeep	nbTrim	...
seq_1	12	8	...
seq_2	11	3	...
seq_3	9	3	...
seq_4	0	12	...

Fig. 5 Schematic representation of the MACSE_ALFIX pipeline. Boxes represent input/output sequence data (blue when unaligned and green when aligned) and are accompanied (on the left) by a small illustrative diagram. On the arrows it is mentioned which subprogram/tool is used and whether this step is optional or not (on/off button). On the right side, additional output files generated are represented in order to provide users with a full traceability picture. The central part of the pipeline, with a colored background, corresponds to the alignment and filtering of the homologous sequences that could be a bottleneck for large datasets

3.4 Aligning Hundreds of Sequences

The computing and memory resources required by MACSE depend on the number and length of the sequences to align. The longest sequence plays a key role in the memory and computation time required by MACSE. When dealing with some long sequences, it may be necessary to significantly increase the memory allocated to the Java virtual machine (using the "-Xmx option"), but the computation time with the default options of the alignSequences subprogram may still be prohibitive. The v2 release of MACSE introduced several options that help balance the computation time and alignment accuracy by limiting the number of alignment refinement steps ("-max_refine_iter") or by gradually narrowing the alignment refinement steps to more local improvements ("-local_realign_init" and "-local_realign_dec" options). As an illustrative example, to build the tenth release of the OrthoMaM database, we had to build more than 20,000 alignments containing up to 116 sequences that could be several Kb long. We designed a pipeline based on MACSE v2 that is well suited for this task. The filtering steps are similar to those of the MACSE_ALFIX pipeline, but the main alignment step here is done by chaining the alignSequences subprogram with MAFFT. The key is to use alignSequences with options that enable MACSE to quite rapidly generate a draft alignment of the coding nucleotide sequences in which potential frameshifts are identified. The resulting amino acids sequences are then aligned using MAFFT, which is much faster than MACSE for aligning fixed amino acid sequences. The drawback of this approach is that some frameshifts may not be as accurately positioned within sequences as they would be with the MACSE_ALFIX pipeline, which may lead HMMCleaner to remove some extra residues. For large datasets of sequences expected to contain few frameshifts, as was the case with the OrthoMaM CDS database, this strategy seems to work remarkably well. The OMM_MACSE pipeline (*see* Fig. 6) has the same mandatory options as the MACSE_ALFIX pipeline:

```
singularity run ./OMM_MACSE_v10.01.sif
                --in_seq_file LOC_48720.fasta
                --out_dir RES_LOC_4872
                --out_file_prefix LOC_48720
```

⇨ *The most simple use case of the OMM_MACSE pipeline for larger datasets.*

Note that, if a dataset contains some pseudogenes or contigs assembled de novo, it may be worth using the refineAlignment subprogram of MACSE to polish the alignment obtained by MAFFT and adjust frameshift positions before applying HMMCleaner.

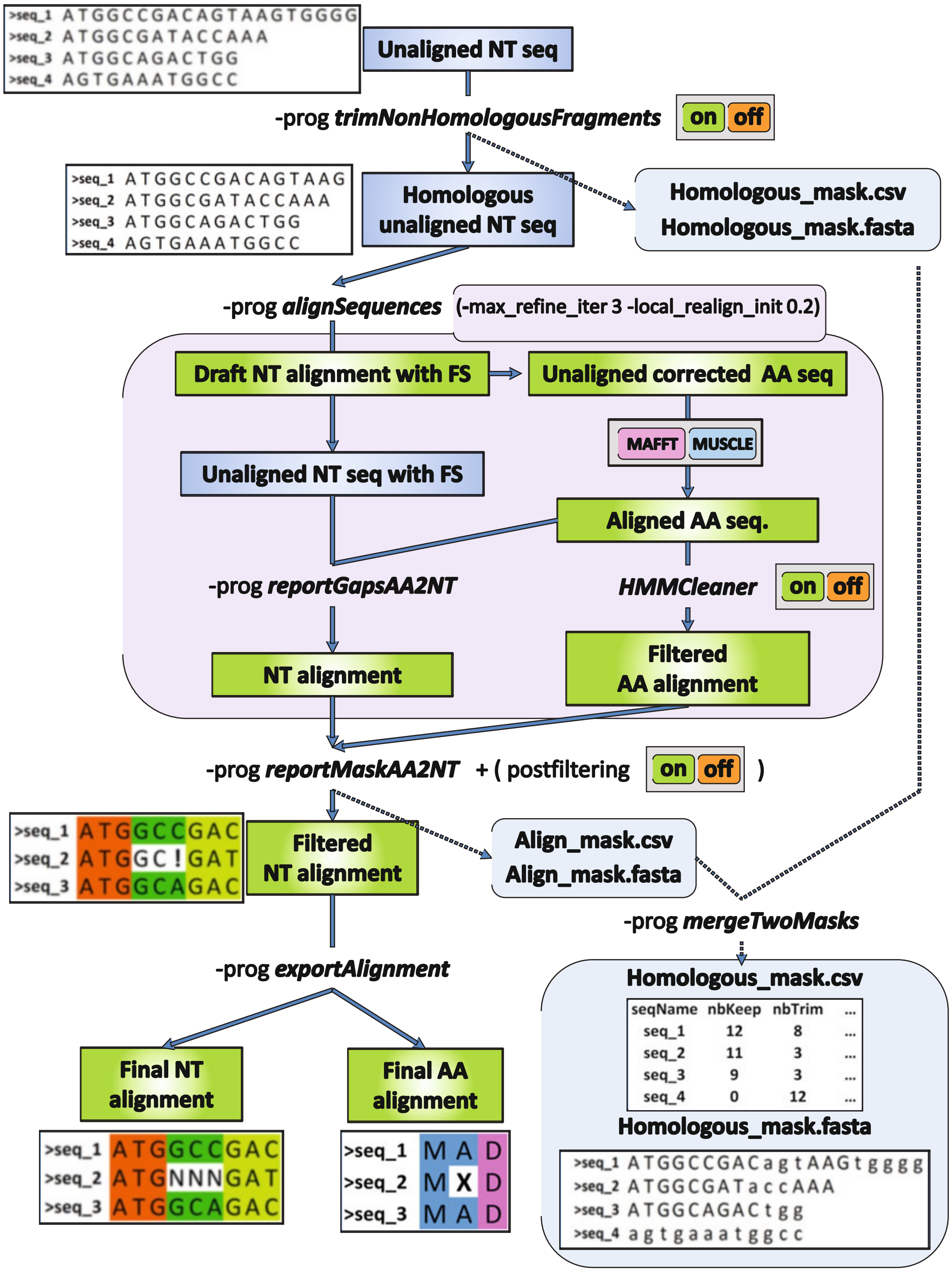


Fig. 6 Schematic representation of the OMM_MACSE pipeline. Boxes represent input/output sequence data (blue when unaligned and green when aligned) and are accompanied, on the left, by a small illustrative diagram. On the arrows it is mentioned which subprogram/tool is used and whether this step is optional or not (on/off button). On the right side, additional output files generated are represented in order to provide users with a full traceability picture. The central part of the pipeline, with a colored background, corresponds to the alignment and filtering of the homologous sequences. This part is the only one that differs from the MACSE_ALFIX pipeline (*see* Fig. 5) and is better suited for large datasets

3.5 Aligning Thousands of Sequences

If you have a very large number of sequences, trying to align them simultaneously is dubious for several technical reasons [16]. It is preferable, as advised by R. Edgar, in the MUSCLE 3.8 [18] user guide (http://www.drive5.com/muscle/muscle_userguide3.8.html), to tackle this problem by leveraging clustering and alignment methods. One possibility is to first build clusters of reasonable size that pool similar sequences (e.g., using UCLUST [19]) in order to align them separately. In a second step, these alignments can be combined to produce the final super-alignment. When there are only two clusters/alignments (e.g., align1.fasta and align2.fasta), they can be aligned with the alignTwoProfiles subprogram of MACSE to produce a single alignment containing all the sequences. This subprogram has many options (mostly the same as alignSequences), but only the options allowing users to specify the two input alignment files (options -p1 and -p2) are mandatory:

java -jar macse.jar ***-prog*** **alignTwoProfiles**
-p1 *align1.fasta*
-p2 *align2.fasta*

⇨ Aligns two previously computed alignments.

When dealing with a handful of clusters, several alignTwoProfiles invocations may be chained to build the global alignment. The idea here is to take the output of one alignTwoProfiles invocation as the p1 profile for the next one. For instance, four alignments can be combined using a MACSE command file as follows:

java -jar macse.jar ***-prog*** *multiPrograms*
-MACSE_command_file *align_multi.macse*

where align_multi.macse is a text file containing this four lines:

-prog alignTwoProfiles -p1 ali1.fasta -p2 ali2.fasta -out_NT ali12.fasta

-prog alignTwoProfiles -p1 ali12.fasta -p2 ali3.fasta -out_NT ali123.fasta

-prog alignTwoProfiles -p1 ali123.fasta -p2 ali4.fasta -out_NT aliAll.fasta

⇨ Basic strategy to align four previously computed alignments.

Using this basic strategy, the final alignment will depend on the order in which the profiles are sequentially added. Under the same rationale as for usual multiple sequence alignment, it would be better to first align the most similar alignments. More elaborate strategies can be designed using MACSE, but this is beyond the scope of this chapter.

3.6 Metabarcoding Applications

Metabarcoding analysis often requires handling thousands of sequences. Such datasets are not directly tractable with the alignSequence subprogram of MACSE, but they can be handled by sequentially adding the newly obtained sequences to a reference alignment containing sequences of related taxa for the targeted barcoding locus (e.g., COX1, matK, rbcL, etc.). We successfully used this approach in the Moorea BIOCODE project on coral reef biodiversity [6].

The initial alignment can be either built from scratch or from an improved version of an existing alignment (using the refineAlignment subprogram of MACSE to unravel some potential sequencing errors/frameshifts). The reference alignment does not need to be huge. For instance, rather than using all available COX1 sequences available in the BOLD database [20], for a given taxonomic group, it may be better to collect some carefully checked sequences that reflect the molecular diversity of the taxonomic groups of interest. Those carefully selected sequences may be aligned using one of the previously detailed strategies (e.g., using the MACSE_ALFIX pipeline). Then, using the enrichAlignment subprogram, problematic reads can be detected while adding the remaining reads to the reference alignment. By default, enrichAlignment adds sequences to an alignment (referred to as the initial alignment) in sequential mode: each sequence is aligned with the current alignment, i.e., that contains the sequences of the initial alignment plus those previously added. Some enrichAlignment options allow users to set thresholds/conditions for a sequence to be discarded and/or to specify that all new sequences must be aligned with the unmodified initial alignment.

The following command line may be used to sequentially enrich an alignment by adding only reads that do not induce too many frameshifts (-maxFS_inSeq), stop codons (-maxSTOP_inSeq) and insertion (maxINS_inSeq) events:

```
java -jar macse.jar -prog enrichAlignment
                    -align Moorea_BIOCODE_small_ref.fasta
                    -seq Moorea_BIOCODE_small_ref.fasta
                    -seq_lr noctural_diet_sample.fasta
                    -gc_def 5
                    -fs_lr 10
                    -stop_lr 10
                    -maxFS_inSeq 0
                    -maxINS_inSeq 0
                    -maxSTOP_inSeq 1
```

⇨ Enrich an initial alignment by conditionally adding sequences to it.

Alternatively, for large datasets, it could be better to work with a fixed alignment (option -fixed_alignment_ON). Working with a fixed alignment is especially convenient when dealing with (meta)-barcoding data since such analyses usually require handling numerous highly similar sequences that are not expected to contain indels. When using this option, all sequences to be added are compared with the same initial alignment. The key advantage is that this allows task parallelization. For example, if there are 50,000 reads/sequences to be added to the initial alignment, this large dataset may be split into 50 sets of 1000 sequences each, and then the tasks may be run in parallel on 50 computers/CPUs. Moreover, if each of the 50,000 sequences can be correctly aligned with the original alignment without inserting gap events in this original alignment, then the aligned version of the 50,000 sequences (that were independently computed) can be merged to the initial alignment to get a valid global alignment.

The enrichAlignment MACSE subprogram not only produces the two usual FASTA output files, respectively, containing the nucleotide and amino acid alignments, but also a tabular text file providing detailed information for each read, including whether it has been added or not and how many stop codons, frameshifts, and insertion events are required to align this read with the reference alignment. This helps to understand why some reads were discarded, to spot reads that have been added but contain few unexpected events (e.g., one internal frameshift) and to compute some overall statistics regarding the input read quality.

4 Conclusion

This chapter describes typical MACSE use cases along with associated command lines and provides two examples of pipelines built from the different MACSE subprograms. In its latest version, MACSE is suitable for bioinformaticians who need to create their own pipelines and for finely controlling the parametering of each subprogram, but it is also accessible to nonspecialists via its graphical interface.

Acknowledgments

We thank François Lechevallier for developing the website hosting MACSE v2, Jimmy Lopez and Khalid Belkhir for deploying MACSE v2 and the OMM_MACSE pipeline as web services on the Montpellier Bioinformatics Platform (MBB). This work was supported by the Agence Nationale de la Recherche "Investissements d'avenir/Bioinformatique" (ANR-10-BINF-01-02 Ancestrome project), Labex CeMEB "Centre Méditerraneen de l'Environnement et de la Biodiversité" (ANR-10-LABX-0004), and the European Research Council (ERC-2015-CoG-683257 ConvergeAnt project). This is contribution ISEM 2020-243 of the Institut des Sciences de l'Evolution de Montpellier.

References

1. Ranwez V, Harispe S, Delsuc F, Douzery EJP (2011) MACSE: Multiple Alignment of Coding SEquences accounting for frameshifts and stop codons. PLoS One 6:e22594
2. Dunn CW, Howison M, Zapata F (2013) Agalma: an automated phylogenomics workflow. BMC Bioinformatics 14:330
3. Yu DW, Ji Y, Emerson BC, Wang X, Ye C, Yang C, Ding Z (2012) Biodiversity soup: metabarcoding of arthropods for rapid biodiversity assessment and biomonitoring. Methods Ecol Evol 3:613–623
4. Ranwez V, Douzery EJP, Cambon C, Chantret N, Delsuc F (2018) MACSE v2: toolkit for the alignment of coding sequences accounting for frameshifts and stop codons. Mol Biol Evol 35:2582–2584
5. Scornavacca C, Belkhir K, Lopez J, Dernat R, Delsuc F, Douzery EJP, Ranwez V (2019) OrthoMaM v10: scaling-up orthologous coding sequence and exon alignments with more than one hundred mammalian genomes. Mol Biol Evol 36:861–862
6. Leray M, Yang JY, Meyer CP, Mills SC, Agudelo N, Ranwez V, Boehm JT, Machida RJ (2013) A new versatile primer set targeting a short fragment of the mitochondrial COI region for metabarcoding metazoan diversity: application for characterizing coral reef fish gut contents. Front Zool 10:34
7. Kurtzer GM, Sochat V, Bauer MW (2017) Singularity: scientific containers for mobility of compute. PLoS One 12:e0177459
8. Gouy M, Guindon S, Gascuel O (2009) SeaView version 4: a multiplatform graphical user interface for sequence alignment and phylogenetic tree building. Mol Biol Evol 27:221–224
9. Larsson A (2014) AliView: a fast and lightweight alignment viewer and editor for large datasets. Bioinformatics 30:3276–3278
10. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30:772–780
11. Wasmuth JD, Blaxter ML (2004) prot4EST: translating expressed sequence tags from neglected genomes. BMC Bioinformatics 5:187
12. Radío S, Fort RS, Garat B, Sotelo-Silveira J, Smircich P (2018) UTRme: a scoring-based tool to annotate untranslated regions in trypanosomatid genomes. Front Genet 9:671
13. Ho-Huu J, Ronfort J, De Mita S, Bataillon T, Hochu I, Weber A, Chantret N (2012) Contrasted patterns of selective pressure in three recent paralogous gene pairs in the Medicago genus (L.). BMC Evol Biol 12:195
14. Singh TR, Tsagkogeorga G, Delsuc F, Blanquart S, Shenkar N, Loya Y, Douzery EJP, Huchon D (2009) Tunicate mitogenomics and phylogenetics: peculiarities of the Herdmania momus mitochondrial genome and support for the new chordate phylogeny. BMC Genomics 10:534
15. Emerling CA, Delsuc F, Nachman MW (2018) Chitinase genes (CHIAs) provide genomic footprints of a post-cretaceous dietary radiation in placental mammals. Sci Adv 4:eaar6478
16. Ranwez V, Chantret N (2020) Strengths and limits of multiple sequence alignment and filtering methods. In " Phylogenetics in the genomic era" edited by Galtier N, Delsuc F, Scornavacca C 2.2:1-2.2-36
17. Di Franco A, Poujol R, Baurain D, Philippe H (2019) Evaluating the usefulness of alignment filtering methods to reduce the impact of errors on evolutionary inferences. BMC Evol Biol 19:21
18. Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucl Acid Res 32:1792–1797
19. Edgar RC (2010) Search and clustering orders of magnitude faster than BLAST. Bioinformatics 26:2460–2461
20. Ratnasingham S, Hebert PD (2007) BOLD: the barcode of life data system (http://www.barcodinglife.org). Mol Ecol Notes 7:355–364

Chapter 5

Cooperation of Spaln and Prrn5 for Construction of Gene-Structure-Aware Multiple Sequence Alignment

Osamu Gotoh

Abstract

Gene-structure-aware multiple sequence alignment (GSA-MSA) is conventionally used as a tool for analyzing evolutionary changes in gene structure, i.e., gain and loss of introns during the course of evolution of homologous eukaryotic genes. Recently, however, it has become apparent that GSA-MSA is a powerful tool for detecting and remedying gene-prediction errors prevalent in genome annotations produced by various genome projects. Unfortunately, the construction of GSA-MSAs has so far required tedious procedures, thereby preventing researchers from enjoying the potential benefits of GSA-MSAs. In this chapter, we introduce a straightforward way for constructing GSA-MSAs when one or more genomic sequences and a set of transcript sequences (protein or full-length cDNAs/CDSs) are given. Our method requires no external tool or extra data, such as annotation files, although a supplementary script can generate a gene-structure-informed (GSI) transcript sequence file from annotation files.

Key words Spliced alignment, Genome mapping, Gene structure, Gene prediction, Multiple sequence alignment, Annotation error, Phylogenetic analysis

1 Introduction

Protein and cDNA sequences subjected to multiple sequence alignment (MSA) analyses are often obtained from sequence databases, many entries of which are derived from gene annotation conducted by various genome projects. However, it has become increasingly apparent that these sequence data contain an appreciable fraction of errors [1–3] due to incomplete genome sequencing/assembly, incorrect gene prediction, mixture of alternatively spliced products, and contamination of pseudogene-derived transcript sequences. The existence of error-harboring sequences not only hampers the construction of MSA itself [4] but also impairs the reliability of downstream analyses, such as phylogenetic reconstruction, functional implication, and sequence-based higher-order structural prediction. Gene-structure-aware multiple sequence alignment (GSA-MSA) has been proven to be a powerful tool for visually detecting these errors [5–7]. However, the existing tools are not necessarily easy to use because in most applications, MSA and a set of gene structural information are prepared and maintained separately, and tedious and error-prone operations are involved in the process. Thus, it is desired to construct GSA-MSAs from genomic and transcript sequences alone without relying on extra data. Two key procedures are essential for this purpose (*see* Fig. 1): (i) a rapid and accurate method for supplementing each transcript sequence with parental gene structural information and (ii) a standalone MSA method that directly utilizes these gene-structure-informed (GSI) transcript sequences as inputs. For the first task, genome mapping and spliced alignment tools, such as Gmap [8], Spaln [9, 10], and Minimap2 [11], can be used, of which Spaln is unique in that not only cDNAs/CDSs but also protein amino acid sequences can be used as queries. Spaln performs genome mapping and spliced alignment seamlessly unlike Scipio [12] and GenBlast [13], which also accept protein queries but require an external genome mapping facility. Compared with other mappers, Spaln is considerably space-efficient, which is beneficial for multi-thread computation. Moreover, Spaln is the most accurate spliced aligner among those developed so far [14]. For the second task, very few programs, such as Malin [15, 16] and Prrn [17], are currently available. Malin is designed to improve concordant intron positions along an externally constructed MSA post-supplemented with gene structural information. In contrast, Prrn is a de novo MSA tool that accepts a set of GSI sequences as input. In this chapter, we introduce Prrn5, the latest version of our MSA program Prrn with a long

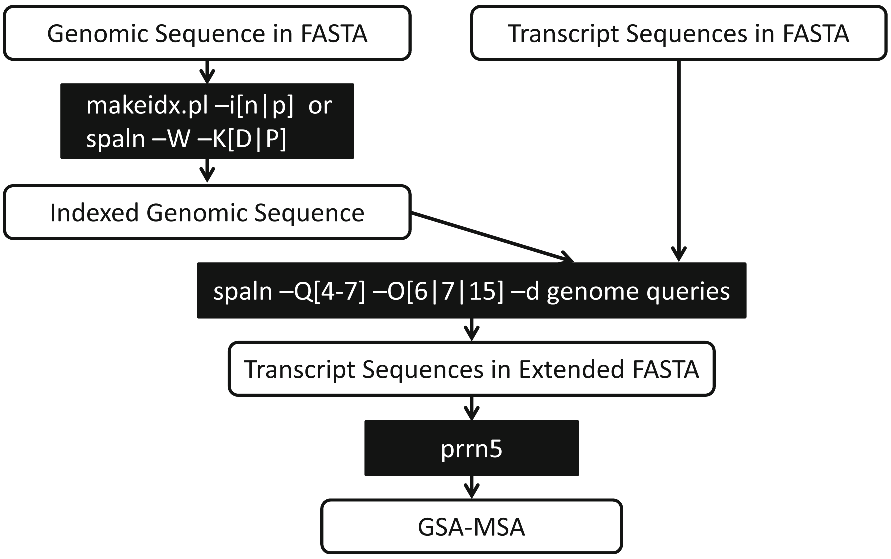

Genomic Sequence in FASTA
Transcript Sequences in FASTA
makeidx.pl –i[n|p] or
spaln –W –K[D|P]
Indexed Genomic Sequence
spaln –Q[4-7] –O[6|7|15] –d genome queries
Transcript Sequences in Extended FASTA
prrn5
GSA-MSA

Fig. 1 Schematic workflow for construction of GSA-MSA from genomic and transcript sequences

history [18, 19]. Compared with its immediate predecessor [17], Prrn5 shows considerably good performance with respect to both speed and accuracy.

2 Methods

2.1 Outline of Spaln Algorithm

2.1.1 Genome Mapping

For protein queries, Spaln first converts the genomic nucleotide sequence into 23-letter Tron (translated codon) codes [20] (*see* **Note 1**). Unlike most other genome mappers that rely on the Blast-like seed and extension strategy [21], Spaln adopts a top-down strategy with multiple phases [9]. In the first phase, Spaln looks for one or several genomic segments that contain the gene(s) homologous to the query. For this purpose, the genomic sequence is divided into blocks of size $B\sqrt{N_G}$ (N_G stands for genome size), and the occurrence of at least one specific k-mer w_k in block b is tabulated and stored in a set of index files. k-mer size k is chosen so that the probability of occurrence of each k-mer in b by chance is much less than 1.0. The tiling (nonoverlapping and contiguous) k-mers on both forward and complementary genomic strands are compiled to make a block index. Upon searching, every k-mer on the query is examined from both ends toward the center for its occurrence in certain blocks, and block score $bs(b)$ of each hit is incremented by word score $s(w_k)$, which depends on the frequency of occurrence of w_k within the entire genomic sequence. For random-sequence queries, $bs(b)$ obeys an extreme value distribution [22]. Using this property, we can identify "significant" blocks that are likely to contain subsequences similar to the query. The genomic segments that are individually sandwiched by the most upstream and most downstream significant blocks within a specified range (*MaxGene*) are passed to the subsequent procedures.

2.1.2 HSP Construction and Chaining

In the second phase, Spaln finds gapless local alignments, i.e., high scoring pairs (HSPs), between each genomic segment and the query based on the local lookup table [23] of spaced seeds [24]. Like Blast2 [25], this table is made from each query at the runtime (*see* **Note 2**). The HSPs are then chained into collinear groups (chained HSPs or cHSPs) with a sparse dynamic programming (DP) algorithm [26]. Multiple cHSPs within one or a few blocks indicate the existence of a tandemly duplicated paralog cluster in which each cHSP corresponds to a gene. The sparse DP algorithm assigns a score to each cHSP. At this stage, weak cHSPs having scores lower than a given threshold are discarded. Spaln keeps best scored at most *N_PARA_CAND* (4 by default) cHSPs that are passed to the subsequent processes. Unlike most other tools, Spaln uses a nonlinear gap penalty function derived from the intron length distribution (ILD) of the relevant or related genome [27] (*see* **Note 3**). This function is useful for sharply discriminating individual members in a tandemly duplicated gene cluster.

Depending on `--Q` option (*see* Subheading 2.2), the above procedure is recursively applied with decreasing seed size down to the specified depth if at least one exon is likely to be missing within the space between neighboring HSPs or either end of the genomic segment.

2.1.3 Merging HSPs to Generate Full Gene Structure

This phase adopts different heuristics depending on the "distance" between neighboring HSPs or an end of the genomic segment (*see* Fig. 2). Note that this phase is skipped when Spaln is invoked in the full DP mode (`--Q0` or `--Q4`). If a pair of neighboring HSPs are contiguous or partially overlap with respect to their query coordinates, the 5′ and 3′ splicing boundaries on the genomic sequence are searched for by a simple linear scan (*see* Fig. 2a). If the space between a pair of neighboring HSPs is positive but too narrow (<

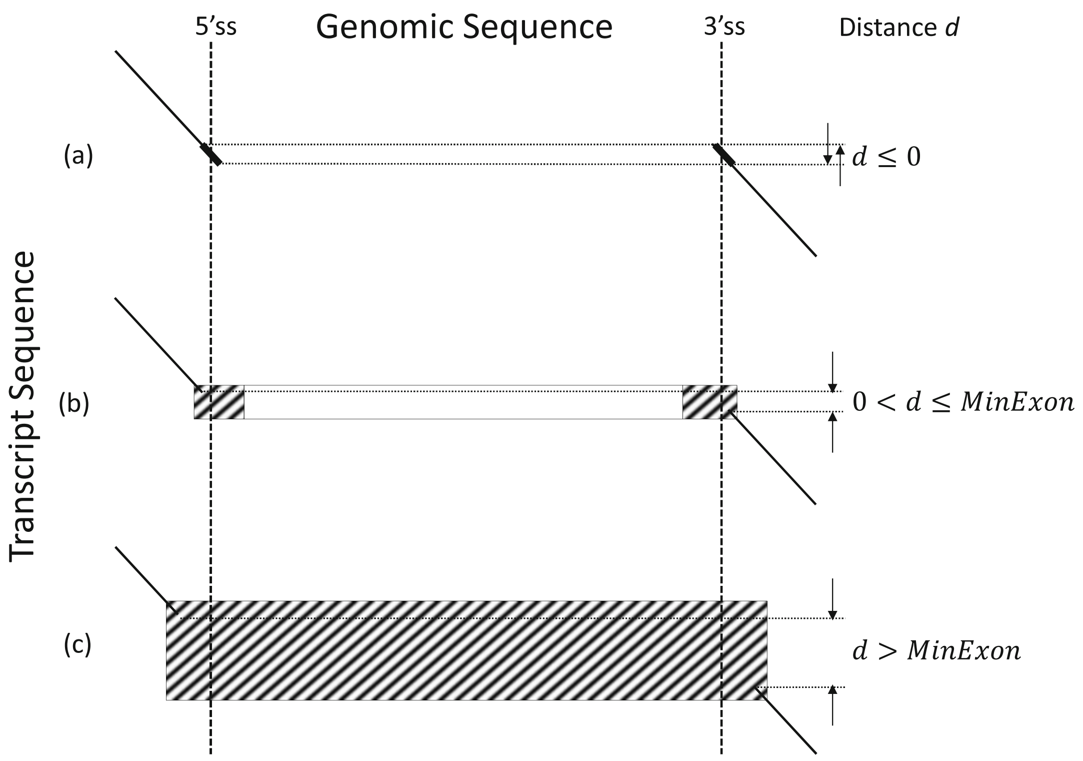


Fig. 2 Three heuristic procedures to obtain final spliced alignment from chained HSPs (diagonal bars). (**a**) The simplest case in which neighboring HSPs contact or overlap with each other along y axis. The 5′ and 3′ splice sites are searched for in the gapless alignments encompassing the junctions (thick bars). (**b**) Skipped spliced alignment under the assumption that no extra exon is present within the genomic region. The DP matrix is filled in the hatched areas only, and the blank area is skipped. (**c**) Standard DP algorithm for spliced alignment within the hatched area

MinExon, which can be reset by `--yE` option) to accommodate an extra exon, a "skipped" DP alignment is performed (*see* Fig. 2b). This is a simplified unidirectional version of the bidirectional "sandwich" or "attack by both sides" algorithm adopted by our previous versions and some other aligners [8, 28]. If neither condition is satisfied, a DP-based spliced alignment method (Subheading 2.1.4) is applied to fill in the space (*see* Fig. 2c).

2.1.4 Spliced Alignment

For spliced alignment, Spaln uses a derivative of the "long gap alignment algorithm" proposed earlier [29] (*see* **Note 4**). Given genomic segment $G = g_1g_2g_3...g_N$ and query $Q = q_1q_2q_3...q_M$, Spaln tries to optimize the following objective function:

$$H(G,Q) = w_A H(S,Q) + w_I \sum_{i\in\{I\}} \gamma_i + w_B \sum_{i\in\{I\}} \delta_i + w_J \sum_{j\in\{J\}} \theta_j \left[+ w_C \sum_{c\in\{C\}} \varphi_c \right], \quad (1)$$

where S denotes either the concatenated inferred exons extracted from the genomic sequence or the inferred amino acid sequence translated therefrom depending on the query type. $H(S, Q)$ is the ordinary alignment score between S and Q with affine [30] or double affine [29] gap penalty, except that frameshifts and immature termination codons negatively contribute to $H(S, Q)$ for protein queries. $\{I\}$, $\{J\}$, and $\{C\}$ denote the sets of inferred introns, inferred exon boundaries, and inferred coding exons, respectively. Intron penalty γ_i is a function of intron length (*see* **Note 3**), whereas δ_i is either 1 or 0 depending on whether the intron position is conserved or not between G and Q if gene structural information of Q is supplied (*see* Subheading 2.2). Exon boundary signal θ_j is calculated from the position-specific mth-order Markov model ($m = 0$, 1, or 2 depending on the number of known exon-intron boundaries). The so-called coding potential φ_c is calculated on the basis of a fifth-order frame-sensitive stationary Markov model trained on known CDSs. Note that this term is included only for protein queries. Two other features, "intron potential" (4- or 5-mer nucleotide composition within an intron) and branch point signal, may be incorporated into the Spaln scoring system [14], but these features are not used by default as their impacts on intron recognition are relatively small [31]. Weight w_A is fixed to 1, whereas other weights w_I, w_J, and w_C together with some other parameter values are empirically determined using a small training set [10]. The bonus given to a conserved intron position, $w_B = 10$, is chosen somewhat arbitrarily. All weight values other than w_A can be reassigned at the runtime by command line options.

The explicit formula for calculating $H(G, Q)$ by induction for a protein query is presented in [20]. The algorithm for a DNA query is simpler than the protein version because we do not need to take care of reading frames. Although it looks quite complicated, the computational cost of the DP algorithm is O(MN). Actually, this cost is reduced by a cutting corners (banded) approximation. With an ordinary traceback procedure, the same order of memory is required. If the expected memory exceeds a certain threshold, previous versions of Spaln are turned into the linear space traceback algorithm [32, 33]. Instead of this bidirectional approach, the latest version of Spaln adopts a unidirectional variant [34] that considerably reduces the code complexity.

2.2 Installation and Execution of Spaln

2.2.1 Installation

The source code of Spaln is available at: http://www.genome.ist.i.kyoto-u.ac.jp/~aln_user/spaln/ or https://github.com/ogotoh/spaln. The binary code executable on a 64-bit Linux system is also available from the former site. Make sure that the three directories, "bin," "table," and "seqdb," are properly installed and accessible. If necessary, set env variables *ALN_TAB* and *ALN_DBS* to indicate the locations of "table" and "seqdb" directories, respectively.

2.2.2 Format of Genomic or Database Sequence

Spaln can be run in different modes (*see* Subheading 2.2.3). To run Spaln in the genome mapping mode, the genomic sequence must be formatted beforehand. This is easily done if the FASTA-format genomic sequence is stored in the "seqdb" directory. Note that the extension of the sequence file must be ".mfa" or ".gf" (*see* **Note 5**). Repeat mask is unnecessary or even harmful if hard-masked. Provided that the file name of the genomic sequence is `genome.gf`, either of the following commands (the second form is valid for version 2.4.0 or later) in the "seqdb" directory will generate a set of necessary index files:

```
$ makeidx.pl -i[n|p|np] [-XGMaxGene] [other options] genome.gf
$ spaln -W -K[D|P] [-XGMaxGene] [other options] genome.gf
```

The arguments `n` (`D`), `p` (`P`), and `np` to `--i` (`--K`) option indicate that `genome.gf` is formatted for query types of DNA, protein, and both of them, respectively, which generates block-index files `genome.bkn`, `genome.bkp` and both of them.

By default, parameter *`MaxGene`* (*see* **Note 6**) is estimated from the file size of `genome.gf`. This, however, can lead to serious underestimation of *`MaxGene`* if `genome.gf` contains only a part of the genomic sequence, e.g., a single chromosome or a supercontig. Although it is possible to define *`MaxGene`* at each runtime, prior definition at the format time is beneficial to prevent careless mistakes at runtime.

To run Spaln in mode 3 or 4 (Subheading 2.2.3), amino acid sequence database `aa_db.faa` should be formatted beforehand with either of the following commands (*see* **Note 7**):

```
$ makeidx.pl -ia [other options] aa_db.faa
$ spaln -W -KA [other options] aa_db.faa
```

To improve sensitivity, four (one contiguous and three patterns of spaced) *k*-mer seeds are used for indexing unless explicitly specified by `--XC` and `--XB` options.

2.2.3 Four Running Modes of Spaln

Spaln can be run in one of the following four modes (*see* **Note 8**):

1. `$ spaln --Q[0-3] [other options] genomic_segment queries`
2. `$ spaln --Q[4-7] --d genome [other options] queries`
3. `$ spaln --Q[4-7] --a aa_db [other options] genomic_segment`
4. `$ spaln --Q[4-7] --a aa_db [other options] aa_queries`

We assume that `genomic_segment` is a FASTA-format sequence file containing only one or a few genes for mode 1, whereas a larger file is feasible for mode 3. There is no particular limit in the size of a `queries` file also in multi-FASTA or extended FASTA format (see below). The molecular type of `queries` is automatically inferred from the residue composition of its first entry. Potential ambiguity can be eliminated by adding the character "D" or "P" to the last argument, e.g., "`nt_queries.fna D`" or "`aa_queries.faa P`" (the quotations are necessary). The arguments `genome` in mode 2 and `aa_db` in modes 3 and 4 are identifiers of the indexed genomic or database sequences as explained in Subheading 2.2.2. Mode 4 performs rapid homology search and (semi-)global alignment of protein sequences (*see* **Note 7**) in a similar manner to that of CD-HIT [35] or Usearch [36].

As for various options, consult with the document attached to the distribution or the Web page mentioned above. Here, only a few that are most relevant to this chapter are discussed.

The argument to `--Q` option, q, determines the running mode of Spaln. If $q \in [0, 3]$, Spaln performs only spliced alignment, whereas $q \in [4, 7]$ invokes a map-and-align mode. Meanwhile, q mod 4 specifies the depth of recursive HSP search, where q mod $4 = 0$ conducts the full DP algorithm without HSP heuristics.

In order to supplement each query with its gene structural information, we should prepare the genomic sequence of the same species as that of the query and run Spaln with `--O6` (cDNA/CDS) or `--O7` (protein) option. The outputs of this run are GSI sequences in the extended FASTA format (*see* Fig. 3) appropriate for subsequent processes. It is very important to note that *these outputs are derived from the genomic sequence and are not necessarily identical to the query sequence* due to polymorphisms,

```
>Oryzsati24117934 Oryzsati1 + [1:2080]  ( 14513218 - 14515297 )
;C join(14513419..14513421,14513996..14514073,14514182..14514362,
;C 14514469..14514566,14514663..14515097)
MARGLKKHLKRLNAPKHWMLDKLGGAFAPKPSSGPHKSRECLPLILIIRNRLKYALTYRE
VISILMQRHVLVDGKVRTDKTYPAGFMDVISIPKTGENYRLLYDTKGRFRLQSVKDEDAK
FKLCKVRSVQFGQKGIPYLNTYDGRTIRYPDPLIKANDTIKIDLETNKIVDFIKFDVGNV
VMVTGGRNTGRVGVIKNREKHKGSFETIHVEDALGHAFATRLGNVFTIGKGNKPWVSLPK
GKGIKLSIIEEQRKRDAAAQAAANA
>Oryzsati24119413 Oryzsati1 + [1:2708]  ( 24480916 - 24483623 )
;C join(24481117..24481264,24481712..24481794,24482122..24482280,
;C 24482771..24482881,24482987..24483053,24483158..24483423)
MVGWRAAGGARAVLRRLSAAAEAAAKQDGRVFAASYSGSSGGVNAPFGLGQYANLLRAQA
FASRGVALNFHQLIRNAGISTTRNLLAADDAMVPVSSPLTPPLGDGEQTDKKGAIVKRLK
VQAIKKDIKQSPKKVNLVAKLVRGMRVEDALLQLQVTVKRAAKTVYQVIHSARANAAHNH
GLDPDKLIVEEAFVGKGLYLKRLSYHAKGRCGVMVRPRCRLTVVVREATAEEEAKIAKLR
VSNYKKLTRKEKQLMPHRLIEVSPRWARKRKEEAGAAA
>Oryzsati24126140 Oryzsati3 - [1:2239]  ( 1234592 - 1232354 )
;C complement(join(1232555..1232760,1233786..1233849,1233947..1234119,
;C 1234206..1234392))
MATTSLSLHGVPSPTATKLSSSFLGAPASFLRPTPPPLAAPSRRALAVRAMAPPKPGGKP
KKVVGLIKLALEAGKATPAPPVGPALGAKGVNIMSFCKEYNAKTAEKAGYIIPVEITVFD
DKSFTFILKTPPASVLLLKAAGIEKGSKEPQREKVGKVTADQVRTIAQEKLPDLNCKSID
SAMRIIAGTAANMGIEVDPPILEKKEKVLL
```

Fig. 3 An example of extended multi-FASTA sequence file. The genomic coordinates of exon boundaries are presented in one or more ";C" lines. The format of the rest of such lines is identical to that in the FEATURE table of the GenBank format. The boundary coordinates are 1-based and inclusive. Besides ";C," any lines starting with ";" as well as blank lines are ignored. A slash "/" at the beginning of a line stops the input of the sequence until the next header line starting with ">." The sequence identifiers after each ">" must be unique within a file

paralog hits, or other causes even though the genomic and query sequences are derived from the same species. By contrast, the output with `--O15` option has the copy of the protein or cDNA query sequence supplemented with inferred gene structural information. This option is useful for obtaining GSI transcript sequences when the genomic sequence of not the cognate but only its closely related species, strain, or individual is available.

Under the default setting for a cDNA/CDS query (`--S3`), Spaln examines both positive and negative (reverse complementary) orientations of the query, and the alignment that gives a better score is adopted. However, if the query possesses a tailing polyA or leading polyT sequence longer than *NA* (12 by default) nucleotides, polyA or polyT is trimmed off, and the orientation is fixed accordingly. The *NA* value is adjustable with `--pa`*NA* option, where `--pa0` disables the abovementioned convention. If the orientation of all the queries is known to be identical, `--S1` or `--S2` option, respectively, fixes the orientation to positive or negative, which can nearly halve the computational time. Although `--S1` or `--S2` option surpasses the polyA- or polyT-based inference, the polyA or polyT sequence is still trimmed off if present. In this connection,`--LS` (local similarity) option is sometimes useful for trimming off weakly matched terminal regions in the alignment.

Although Spaln is not highly sensitive to small changes in parameter values, the proper selection of the species-specific parameter set with `--T` option is essential for accurate gene structure prediction; *excessively fragmented exon prediction is a sign of improper choice of parameter set*. Currently, the "gnm2tab" file in the "table' directory lists 105 parameter sets useful for approximately 700 species. These numbers will be at least doubled in near future. Even though the exact same species is not found in the list, the choice of the evolutionarily closest species in the list may work fine. For example, `–T Magnolio` may be useful for most grass genomes.

If both cognate pairs of genomic and transcript sequences are of high quality, i.e., there are only very few mismatches and indels are expected in their alignments, `–yX0` (intra species) option may be helpful, which adopts severer mismatch and gap penalties and is more aggressive in finding missing exons, such as mini, micro, or terminal exons, than the default setting.

Finally, with `--M`*m*`.`*n* option, we can specify the number of outputs per query (*m*, default = 1) and the *N_PARA_CAND* value (*n*, default = 4) described in Subheading 2.1.2. In the earlier versions of Spaln, the *m*-best outputs were chosen on the basis of the cHSP scores (*see* Subheading 2.1.2). After version 2.4.0, however, the final alignment scores are used to sort *N_PARA_CAND* candidate genes. This change in the algorithm has considerably increased the chance of finding the cognate gene rather than its paralogs including processed (pseudo-)genes.

2.2.4 Examples

To acquire some ideas about the performance of Spaln, we ran the following commands to format the human genomic sequence, homosapi_g.gf (GRCh38.p11, 3.233Gb):

```
$ spaln -W -KD -t5 homosapi_g.gf
$ spaln -W -KP -t5 homosapi_g.gf
```

The formatting took 3.95 and 5.93 min, respectively, on our machine (CentOS 6.2 equipped with 64 GB memory, 2 CPUs of Xeon E5-2687W (16 cores) 3.10 GHz, and 2.0 TB storage). Increasing the thread number specified by `--t` option did not further shorten the formatting time. (The multi-thread option at the format time is supported from version 2.4.0).

Then, we mapped and aligned the whole human RefSeq sequences (ftp://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/mRNA_Prot/) with the following commands:

```
$ spaln -Q7 -O6 -S1 -pa3 -d homosapi_g -T Tetrapod -t16 -pq
refseq_n.fna
$ spaln -Q7 -O7 -d homosapi_g -T Tetrapod -t16 -yX0 -pq
refseq_a.faa
```

Table 1
Summary of results of Spaln for mapping human RefSeq sequences on human genome

Query	# Seq.	Total residues	Mapped[a]	Exact match[b]	<1% mismatch[c]	Run time[d] (min)	Max memory[e] (GB)
DNA	159104	572Mbp	158463	154944	158375	3.48	7.90
%	100		99.597	97.38	99.54		
Protein	112481	6017kaa	112474	10651	111352	16.05	23.33
%	100		99.993	94.69	99.00		

[a]This number includes queries mapped to potentially noncognate (paralogous) sites
[b]Number of sequences for each of which output GSI sequence is identical to original query sequence. PolyA tails are removed upon sequence comparison
[c]Number of sequences for each of which output GSI sequence matches original query sequence with less than 1% mismatch or indels. PolyA tails are removed upon sequence comparison
[d]Elapsed time measured with /usr/bin/time –v command
[e]Maximum resident set size measured with /usr/bin/time –v command (due to a bug in the version of /usr/bin/time we used, the listed values are approximately four times larger than the real memory consumption)

`--pq` (quiet) option suppressed the warning messages output to *stderr*. The results are summarized in Table 1. Almost all the unmapped sequences were shorter than the lower limits (72 bp for DNA and 36 aa for protein) predefined by Spaln. Most alignment errors for protein queries originated from the incorrect identification of the first or the last coding exon shorter than the word size used in the first phase. This issue will be addressed in future updates.

2.3 Outline of Prrn5 Algorithm

The strategy by which Prrn [18, 19] constructs an MSA is now a standard one consisting of (1) calculation of a distance matrix, (2) calculation of a guide tree, (3) progressive alignment, (4) calculation of pair weights, (5) iterative refinement, and, (6) if necessary, iteration of 1–5 until convergence or by a pre-specified number of times. Since version 4, Prrn can incorporate genome structural information associated with each member. The outline of the Prrn5 algorithm is illustrated in Fig. 4. Here, only some features new to Prrn5 are briefly discussed.

2.3.1 Guide Tree or Guide Forest

Unlike previous versions of Prrn as well as most other MSA methods, Prrn5 does not try to calculate all elements of the distance matrix but fills in only limited matrix elements close to the diagonal (*see* Fig. 4a). Let M be the number of input sequences and Prox $(M) < M$ be a user-defined number (by default Prox$(M) = \sqrt{M}$ + a small constant). Then, for each member $i \in [1, M]$, (semi-)global alignments between i and $J \leq$ Prox(M) other members (j) presumably most similar to i are calculated in much the same way as that of mode 4 discussed in Subheading 2.2.4. The block scores of these selected members must exceed a certain threshold so that J can be smaller than *Prox*(M). It should also be noted that these J members

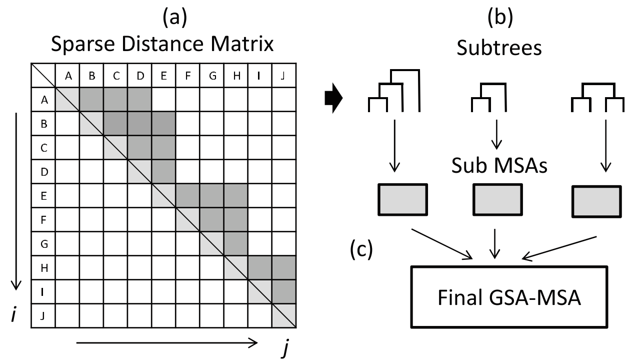


Fig. 4 Outline of Prrn5 algorithm. (**a**) Sparse distance matrix where only shaded squares have distance values. (**b**) Guide subtrees and sub-alignments calculated therefrom. (**c**) Multiple MSA alignment that merges sub-alignments

are selected on the basis of the block score rather than the final alignment score. Alignments between members $i \leq j$ are skipped to fill only the upper right part of the distance matrix. The total number of $E \approx M\overline{J}/2$ ($\overline{J}$ denotes the average of Js) filled elements in the resulting sparse distance matrix represents the edges that connect similar members (nodes), i and j, in an undirected graph. The edge weight (distance) between a pair of connected members is calculated from the corresponding pairwise alignment.

A single-linkage guide tree is obtained by Kruskal's algorithm [37], and an MSA is constructed in a standard way. However, when the generated graph is not connected due to the sparsity of the distance matrix, a forest consisting of several subtrees is obtained (*see* Fig. 4b). In such a case, sub-alignments corresponding to individual subtrees are constructed in the standard way. These sub-alignments are finally joined into a large MSA by a multiple MSA (or profile) alignment method (*see* Fig. 4c). The worst-case computational cost of Kruskal's algorithm is O(E log (E)) [37], which is considerably smaller than that of the more popular UPGMA and neighbor-joining methods of O(M^2)~O(M^3). Moreover, several research groups have reported that the single-linkage tree consistently outperforms UPGMA or neighbor-joining tree when used as a guide tree [38, 39].

2.3.2 Objective Function and Group-to-Group Alignment

The objective function of Prrn is the weighted sum-of-pairs (WSP) score [40]. At each step of a progressive or an iterative procedure, Prrn attempts to obtain the optimal pairwise alignment between input sequences or pre-aligned groups of sequences. Considering the balance between speed and accuracy, Prrn5 adopts Algorithm C, which is less rigorous but faster than previously used Algorithm D [41, 42]. Optionally, simpler Algorithm B, which is nearly equivalent to those used in Mafft [43] and Muscle [44], can be used. Although Algorithm C is more costly than

Algorithm B, Algorithm C generally requires a smaller number of iterations until convergence than Algorithm B.

Let $w_{i,\ j}$ be the pair weight for members i and j. An intron position conserved between i and j is given bonus $w_{i,\ j}w_B$, where w_B is equivalent to that defined in Eq. (1). "Position" is measured on the basis of the coding nucleotide sequence, so that the phase of the intron within a codon (phase = 1 or 2) or between codons (phase = 0) is significant. No penalty is assigned to nonconserved intron positions. The intrinsically fuzzy nature of phase-0 intron position within a gapped region [15] may be unresolvable at the progressive alignment phase but resolvable at the iterative refinement phase. Intron positions shared by plural members are stored in profile form, which enables fast calculation of the bonus value in sequence-to-group or group-to-group alignment procedures.

2.4 Installation and Execution of Prrn5

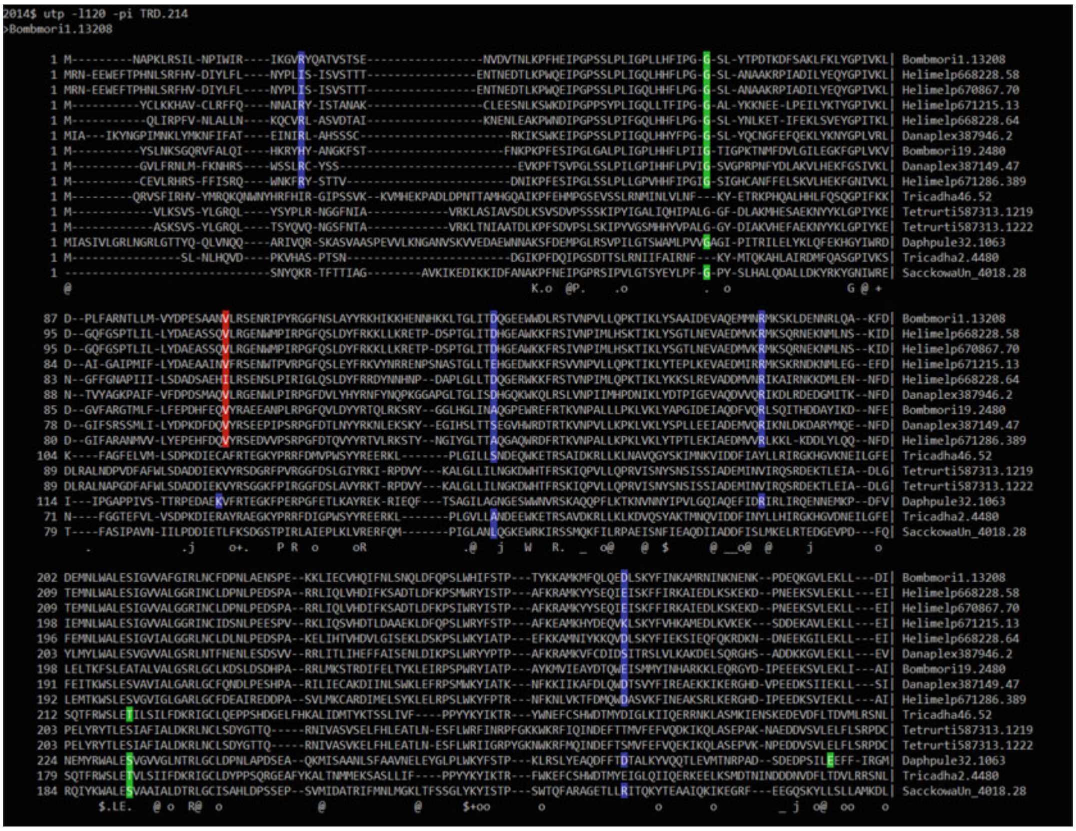
The source codes of Prrn5 together with some associated programs, such as Aln, Utn, Utp, and Anno2gsiseq.pl, are available at http://www.genome.ist.i.kyoto-u.ac.jp/~aln_user/prrn/ or https://github.com/ogotoh/prrn_aln. If necessary, set env variables *ALN_TAB* to indicate the locations of the "table" directory, which is shared by Spaln. All programs including Spaln are likely to run on any Unix/Linux system including WSL (Windows Subsystem for Linux) and MacOS. As compilation and installation are straightforward, no further explanation is given here.

It is also very easy to run Prrn5. Let `Input.mfa` be a sequence file in the extended FASTA format (*see* Fig. 3). `Input.mfa` may be obtained as an output from Spaln with `--O6`, `--O7`, or `--O15` option. Alternatively, desired GSI sequences may be generated by using Anno2gsiseq.pl script (*see* **Note 9**). Several multiple- (.mfa) and single-sequence (.fa) files may be combined by simply enumerating them as the command line arguments such that.

```
$ prrn5 [options] Input1.mfa Input2.mfa Seq1.fa Seq2.fa
```

The input order is not significant except that prrn5 is invoked with `--U` option (update mode, **Note 10**), in which the first argument must be the MSA file to be updated. The output of the above command goes to *stdout*. To explicitly specify the output file, `-o output.prrn5` option is usable. If an input file contains pre-aligned MSA as judged from the existence of at least one internal gap, that alignment is internally frozen unless `--U` option is specified. This is the simplest way to perform multiple MSA alignment.

To visualize the distribution of intron positions along MSA, use `--pi` option (*see* Fig. 5). Alternatively, `-ph` option produces a simple html file that may be visualized by any favorite html viewer. To review the intron position profile of an existing Prrn5 output, use



```
2014$ utp -l120 -pi TRD.214
>Bombmori1.13208

  1 M---------NAPKLRSIL-NPIWIR----IKGVRYQATVSTSE-----------------NVDVTNLKPFHEIPGPSSLPLIGPLLHFIPG-G-SL-YTPDTKDFSAKLFKLYGPIVKL| Bombmori1.13208
  1 MRN-EEWEFTPHNLSRFHV-DIYLFL----NYPLIS-ISVSTTT----------------ENTNEDTLKPWQEIPGPSSLPLIGQLHHFLPG-G-SL-ANAAKRPIADILYEQYGPIVKL| Helimelp668228.58
  1 MRN-EEWEFTPHNLSRFHV-DIYLFL----NYPLIS-ISVSTTT----------------ENTNEDTLKPWQEIPGPSSLPLIGQLHHFLPG-G-SL-ANAAKRPIADILYEQYGPIVKL| Helimelp670867.70
  1 M----------YCLKKHAV-CLRFFQ----NNAIRY-ISTANAK-----------------CLEESNLKSWKDIPGPPSYPLIGQLLTFIPG-G-AL-YKKNEE-LPEILYKTYGPIVKL| Helimelp671215.13
  1 M-----------QLIRPFV-NLALLN----KQCVRL-ASVDTAI-----------------KNENLEAKPWNDIPGPSSLPIFGQLHHFLPG-G-SL-YNLKET-IFEKLSVEYGPITKL| Helimelp668228.64
  1 MIA---IKYNGPIMNKLYMKNFIFAT----EINIRL-AHSSSC----------------------RKIKSWKEIPGPSSLPIIGQLHHYFPG-G-SL-YQCNGFEFQEKLYKNYGPLVRL| Danaplex387946.2
  1 M----------YSLNKSGQRVFALQI----HKRYHY-ANGKFST--------------------FNKPKPFESIPGLGALPLIGPLHHFLPIIG-TIGPKTNMFDVLGILEGKFGPLVKV| Bombmori19.2480
  1 M----------GVLFRNLM-FKNHRS----WSSLRC-YSS--------------------------EVKPFTSVPGLSSLPILGPIHHFLPVIG-SVGPRPNFYDLAKVLHEKFGSIVKL| Danaplex387149.47
  1 M----------CEVLRHRS-FFISRQ----WNKFRY-STTV------------------------DNIKPFESIPGLSSLPLLGPVHHFIPGIG-SIGHCANFFELSKVLHEKFGNIVKL| Helimelp671286.389
  1 M---------QRVSFIRHV-YMRQKQNWNYHRFHIR-GIPSSVK--KVMHEKPADLDPNTTAMHGQAIKPFEHMPGSEVSSLRNMINLVLNF---KY-ETRKPHQALHHLFQSQGPIFKK| Tricadha46.52
  1 M------------VLKSVS-YLGRQL----YSYPLR-NGGFNIA------------VRKLASIAVSDLKSVSDVPSSSKIPYIGALIQHIPALG-GF-DLAKMHESAEKNYYKLGPIYKE| Tetrurti587313.1219
  1 M------------ASKSVS-YLGRQL----TSYQVQ-NGSFNTA------------VRKLTNIAATDLKPFSDVPSLSKIPYVGSMHHYVPALG-GY-DIAKVHEFAEKNYYKLGPIYKE| Tetrurti587313.1222
  1 MIASIVLGRLNGRLGTTYQ-QLVNQQ----ARIVQR-SKASVAASPEVVLKNGANVSKVVEDAEWNNAKSFDEMPGLRSVPILGTSWAMLPVVGAGI-PITRILELYKLQFEKHGYIWRD| Daphpule32.1063
  1 M----------------SL-NLHQVD----PKVHAS-PTSN------------------------DGIKPFDQIPGSDTTSLRNIIFAIRNF---KY-MTQKAHLAIRDMFQASGPIVKS| Tricadha2.4480
  1 ------------------------------SNYQKR-TFTTIAG--------AVKIKEDIKKIDFANAKPFNEIPGPRSIPVLGTSYEYLPF-G-PY-SLHALQDALLDKYRKYGNIWRE| SacckowaUn_4018.28
    @                                                               K.o  @P.    .o          .  o            G @ +

 87 D--PLFARNTLLM-VYDPESAANVLRSENRIPYRGGFNSLAYYRKHIKKHENNHKKLTGLITDQGEEWWDLRSTVNPVLLQPKTIKLYSAAIDEVAQEMMNRMKSKLDENNRLQA--KFD| Bombmori1.13208
 95 D--GQFGSPTLIL-LYDAEASSQVLRGENWMPIRPGFQSLDYFRKKLLKRETP-DSPTGLITDHGEAWKKFRSIVNPIMLHSKTIKLYSGTLNEVAEDMVKRMKSQRNEKNMLNS--KID| Helimelp668228.58
 95 D--GQFGSPTLIL-LYDAEASSQVLRGENWMPIRPGFQSLDYFRKKLLKRETP-DSPTGLITDHGEAWKKFRSIVNPIMLHSKTIKLYSGTLNEVAEDMVKRMKSQRNEKNMLNS--KID| Helimelp670867.70
 84 D--AI-GAIPMIF-LYDAEAAINVFRSENWTPVRPGFQSLEYFRKVYNRRENPSNASTGLLTEHGEDWKKFRSVVNPVLLQPKTIKLYTEPLKEVAEDMIRRMKSKRNDKNMLEG--EFD| Helimelp671215.13
 83 N--GFFGNAPIII-LSDADSAEHILRSENSLPIRIGLQSLDYFRRDYNNHNP--DAPLGLLTDQGERWKKFRSTVNPIMLQPKTIKLYKKSLREVADDMVNRIKAIRNKKDMLEN--NFD| Helimelp668228.64
 88 N--TVYAGKPAIF-VFDPDSMAQVLRGENWLPIRPGFDVLYHYRNFYNQPKGGAPGLTGLISDHGQKWKQLRSLVNPIIMHPDNIKLYDTPIGEVAQDVVQRIKDLRDEDGMITK--NFD| Danaplex387946.2
 85 D--GVFARGTMLF-LFEPDHFEQVYRAEEANPLRPGFQVLDYYRTQLRKSRY--GGLHGLINAQGPEWREFRTKVNPALLLPKLVKLYAPGIDEIAQDFVQRLSQITHDDAYIKD--NFE| Bombmori19.2480
 78 D--GIFSRSSMLI-LYDPKDFDQVYRSEEPIPSRPGFDTLNYYRKNLEKSKY--EGIHSLTTSEGVHWRDTRTKVNPVLLKPKLVKLYSPLLEEIADEMVQRIKNLDKDARYMQE--NFD| Danaplex387149.47
 80 D--GIFARANMVV-LYEPEHFDQVYRSEDVVPSRPGFDTQVYYRTVLRKSTY--NGIYGLTTAQGAQWRDFRTKVNPALLKPKLVKLYTPTLEKIAEDMVVRLKKL-KDDLYLQQ--NFD| Helimelp671286.389
104 K----FAGFELVM-LSDPKDIECAFRTEGKYPRRFDMVPWSYYREERKL-------PLGILLSNDEQWKETRSAIDKRLLKLNAVQGYSKIMNKVIDDFIAYLLRIRGKHGVKNEILGFE| Tricadha46.52
 89 DLRALNDPVDFAFWLSDADDIEKVYRSDGRFPVRGGFDSLGIYRKI-RPDVY---KALGLLILNGKDWHTFRSKIQPVLLQPRVISNYSNSISSIADEMINVIRQSRDEKTLEIA--DLG| Tetrurti587313.1219
 89 DLRALNAPGDFAFWLSDADDIEKVYRSGGKFPIRGGFDSLAVYRKT-RPDVY---KALGLLILNGKDWHTFRSKIQPVLLQPRVISNYSNSISSIADEMINVIRQSRDEKTLEIA--DLG| Tetrurti587313.1222
114 I---IPGAPPIVS-TTRPEDAEKVFRTEGKFPERPGFETLKAYREK-RIEQF---TSAGILAGNGESWWNVRSKAQQPFLKTKNVNNYIPVLGQIAQEFIDRIRLIRQENNEMKP--DFV| Daphpule32.1063
 71 N----FGGTEFVL-VSDPKDIERAYRAEGKYPRRFDIGPWSYYREERKL-------PLGVLLANDEEWKETRSAVDKRLLKLKDVQSYAKTMNQVIDDFINYLLHIRGKHGVDNEILGFE| Tricadha2.4480
 79 T----FASIPAVN-IILPDDIETLFKSDGSTPIRLAIEPLKLVRERFQM-------PIGLANLQGKEWRKIRSSMQKFILRPAEISNFIEAQDIIADDFISLMKELRTEDGEVPD---FQ| SacckowaUn_4018.28
      .              .j     o+.    P R  o    oR                 .@  j   W   R.  _  o@    @ $     @ __o@  @     j        o

202 DEMNLWALESIGVVAFGIRLNCFDPNLAENSPE--KKLIECVHQIFNLSNQLDFQPSLWHIFSTP---TYKKAMKMFQLQEDLSKYFINKAMRNINKNENK--PDEQKGVLEKLL---DI| Bombmori1.13208
209 TEMNLWALESIGVVALGGRINCLDPNLPEDSPA--RRLIQLVHDIFKSADTLDFKPSMWRYISTP---AFKRAMKYYSEQIEISKFFIRKAIEDLKSKEKD--PNEEKSVLEKLL---EI| Helimelp668228.58
209 TEMNLWALESIGVVALGGRINCLDPNLPEDSPA--RRLIQLVHDIFKSADTLDFKPSIWRYISTP---AFKRAMKYYSEQIEISKFFIRKAIEDLKSKEKD--PNEEKSVLEKLL---EI| Helimelp670867.70
198 IEMNLWALESIGVVALGGRINCIDSNLPEESPV--RKLIQSVHDTLDAAEKLDFQPSLWRYFSTP---AFKEAMKHYDEQVKLSKYFVHKAMEDLKVKEK---SDDEKAVLEKLL---EI| Helimelp671215.13
196 FEMNLWALESIGVIALGGRLNCLDLNLPEDSPA--KELIHTVHDVLGISEKLDSKPSLWKYIATP---EFKKAMNIYKKQVDLSKYFIEKSIEQFQKRDKN--DNEEKGILEKLL---EI| Helimelp668228.64
203 YLMYLWALESVGVVALGSRLNTFNENLESDSVV--RRLITLIHEFFAISENLDIKPSLWRYYPTP---AFKRAMKVFCDIDSITRSLVLKAKDELSQRGHS--ADDKKGVLEKLL---EV| Danaplex387946.2
198 LELTKFSLEATALVALGSRLGCLKDSLDSDHPA--RRLMKSTRDIFELTYKLEIRPSPWRYIATP---AYKMVIEAYDTQWEISMMYINHARKKLEQRGYD-IPEEEKSVLEKLI---AI| Bombmori19.2480
191 FEITKWSLESVAVIALGARLGCFQNDLPESHPA--RILIECAKDIINLSWKLEFRPSMWKYIATK---NFKKIIKAFDLQWDTSVYFIREAEKKIKERGHD-VPEEDKSIIEKLL---SI| Danaplex387149.47
192 LEMTKWSLESVGVIGLGARLGCFDEAIREDDPA--SVLMKCARDIMELSYKLELRPSLWKYFPTR---NFKNLVKTFDMQWDASVKFINEAKSRLKERGHD-IPEEDKSVIEKLL---AI| Helimelp671286.389
212 SQTFRWSLETILSILFDKRIGCLQEPPSHDGELFHKALIDMTYKTSSLIVF----PPYYKYIKTR---YWNEFCSHWDTMYDIGLKIIQERRNKLASMKIENSKEDEVDFLTDVMLRSNL| Tricadha46.52
203 PELYRYTLESIAFIALDKRLNCLSDYGTTQ-----RNIVASVSELFHLEATLN-ESFLWRFINRPFGKKWKRFIQINDEFTTMVFEFVQDKIKQLASEPAK-NAEDDVSVLELFLSRPDC| Tetrurti587313.1219
203 PELYRYTLESIAFIALDKRLNCLSDYGTTQ-----RNIVASVKELFHLEATLN-ESFLWRIIGRPYGKNWKRFMQINDEFTSMVFEFVQEKIKQLASEPVK-NPEDDVSVLELFLSRPDC| Tetrurti587313.1222
224 NEMYRWALESVGVVGLNTRLGCLDPNLAPDSEA--QKMISAANLSFAAVNELEYGLPLWKYFSTP---KLRSLYEAQDFFTDTALKYVQQTLEVMTNRPAD--SDEDPSILEEFF-IRGM| Daphpule32.1063
179 SQTFRWSLETVLSIIFDKRIGCLDYPPSQRGEAFYKALTNMMEKSASLLIF----PPYYKYIKTR---FWKEFCSHWDTMYEIGLQIIQERKEELKSMDTNINDDDNVDFLTDVLRRSNL| Tricadha2.4480
184 RQIYKWALESVAAIALDTRLGCISAHLDPSSEP--SVMIDATRIFMNLMGKLTFSSGLYKYISTP---SWTQFARAGETLLRITQKYTEAAIQKIKEGRF---EEGQSKYLLSLLAMKDL| SacckowaUn_4018.28
           $.LE.   @ o  R@  o               @             @     $+oo      o               o       o          _ j  o@  oo    o
```

Fig. 5 Screen dump of an output from Utp with –pi option. Intron positions and phases are indicated by colors. Red: phase 0 intron immediately before the colored residue. Green and blue: phases 1 and 2 introns within colored residues, respectively

```
$ prrn5 -I0 -p[i|h] output.prrn5
```

The same results are obtained by Utn (DNA) or Utp (protein) command (*see* **Note 11**) such that

```
$ utn (or utp) -l -p[i|h] output.prrn5
```

`--I`$m.n$ option to Prrn5 specifies the maximum numbers of outer (m) and inner (n) iteration loops, respectively, corresponding to stage 6 and stage 5 in the algorithm shown at the top of this section. When $m = 0$, only progressive alignment is performed, although no alignment action is taken if the input is pre-aligned and `--U` option is not specified. If n is unspecified, the inner loop is continued until convergence. The default values are $m = 1$ and $n = \infty$.

Our primary motivation for developing a GSA-MSA method was to use it as a component of tools Refgs.pl and Refgs3.pl [17] for detecting and correcting gene-prediction errors prevalent in various genome projects. We are now developing another tool in this line for trans-species-wide prediction of protein-coding genes belonging to a specific family without relying on existing annotations. Undoubtedly, Spaln and Prrn5 discussed in this chapter play the most important roles in that attempt.

3 Notes

1. A Tron sequence simultaneously represents the original genomic nucleotide sequence and the conceptual amino acid sequence translated therefrom in three frames [20]. The 23-letter Tron code is compatible with nearly all standard and nonstandard nuclear genetic codes, although a nonstandard genetic code must be specified by `--C`*N* option, where *N* stands for the "transl_table" number defined by NCBI (https://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/index.cgi?chapter=cgencodes).
2. A local lookup table can also be made from the genomic sequence at the format time if `--E` option is specified. However, this table requires large disk space and memory. Hence, `–E` option would be useful only when your system is equipped with large storage and memory.
3. Each species-specific ILD is compactly and accurately modelled by a superposition of 1–3 Frechet distributions [27]. Currently available ILDs are listed in IldModel.txt in the "table" directory.
4. Early spliced alignment algorithms such as Procrustes [45] relied on the combinatory optimization of predefined exon candidates. This approach is not efficient because the sparse DP algorithm does not naturally take advantage of the collinear nature of the alignment between genomic and transcript sequences, apart from the difficulty in enumerating a necessary and sufficient number of plausible exon candidates. Hence, a majority of spliced alignment methods developed so far, namely, Nap [46], EST_Genome [47], GeneSeqer [48], Aln [20], Exalin [49], and many others, have the same basic structure in which ordinary pairwise alignment is extended to incorporate long gaps, the ends of which should conform to known splice site consensuses.

5. This restriction comes from the requirement of using the "make" command. As the latest version of Spaln can bypass the use of make command, this restriction is no longer obligatory. For example, the next command is equivalent to `makeidx.pl --in genome.gf` if `genome.fna` is an alias of `genome.gf`:

   `$ spaln --W --KD genome.fna`

   However, consistent use of extensions is beneficial to reduce accidental errors.
6. *MaxGene* is defined in bp unit, whereas postfix K or M may be used to indicate kilo or mega bp. For example, `--XG2400K` is appropriate for most mammals, whereas `--XG2.4 M` is legitimate or not depending on the version of Spaln.
7. Actually, this step can be bypassed if Spaln is executed in the following way in place of mode 4 in Subheading 2.2.3:

```
$ spaln -Q[4-7] [other options] aa_db.faa aa_queries
```

   The first argument represents the database that is internally formatted in the same way as that described in Subheading 2.2.2. This onetime index information is discarded at the end of each run.
8. When `--d` (or `--a`) option is replaced with `--D` (or `--A`) option, the entire genomic (or database) sequence is read into the memory. This option demands larger internal memory than usual but somewhat accelerates overall computational time, especially when the genomic (or database) sequence file is stored in the gzipped form in the disk.
9. Anno2gsiseq.pl takes two arguments, gff/gtf file and transcript sequence file, in addition to `--f` option (e.g., `–f NCBI` or `--f EMBL`) that specifies the dialect of the gff/gtf file.
10. `--U` option is primarily used in the iterative refinement of predicted gene structures [17]. With this option, the pertinent members in the old MSA specified as the first argument are replaced by newly supplied GSI sequences of the same identifiers specified by the following arguments. The modified MSA is then iteratively refined as stage 5 described near the beginning of Subheading 2.3. New sequences without pertinent members in the old MSA are added to the old MSA, whose internal alignment is unfrozen, and then iterative refinement is performed. Thus, `--U` option is also used as the command to unfrozen existing MSA(s).
11. Utn (or Utp) can also be used to display intron position information in several ways, such as presence or absence binary matrix (`--B0`), list of column positions along MSA (`--B1`), and so on. Use `--h` option for more information.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number 16H06279 (PAGS) granted to Prof. Kiyoshi Asai (The University of Tokyo). The author thanks Prof. Kiyoshi Asai and Dr. Toutai Mitsuyama (AIST) for their support and encouragement of this work.

References

1. Nagy A, Hegyi H, Farkas K, Tordai H, Kozma E, Banyai L, Patthy L (2008) Identification and correction of abnormal, incomplete and mispredicted proteins in public databases. BMC Bioinformatics 9:353. https://doi.org/10.1186/1471-2105-9-353
2. Prosdocimi F, Linard B, Pontarotti P, Poch O, Thompson JD (2012) Controversies in modern evolutionary biology: the imperative for error detection and quality control. BMC Genomics 13:5. https://doi.org/10.1186/1471-2164-13-5
3. Patthy L (2016) Identification and correction of erroneous protein sequences in public databases. Methods Mol Biol 1415:179–192. https://doi.org/10.1007/978-1-4939-3572-7_9
4. Katoh K, Standley DM (2016) A simple method to control over-alignment in the MAFFT multiple sequence alignment program. Bioinformatics 32(13):1933–1942. https://doi.org/10.1093/bioinformatics/btw108
5. Fawal N, Savelli B, Dunand C, Mathe C (2012) GECA: a fast tool for gene evolution and conservation analysis in eukaryotic protein families. Bioinformatics 28(10):1398–1399. https://doi.org/10.1093/bioinformatics/bts153
6. Hammesfahr B, Odronitz F, Muhlhausen S, Waack S, Kollmar M (2013) GenePainter: a fast tool for aligning gene structures of eukaryotic protein families, visualizing the alignments and mapping gene structures onto protein structures. BMC Bioinformatics 14:77. https://doi.org/10.1186/1471-2105-14-77
7. Wilkerson MD, Ru Y, Brendel VP (2009) Common introns within orthologous genes: software and application to plants. Brief Bioinform 10(6):631–644. https://doi.org/10.1093/bib/bbp051
8. Wu TD, Watanabe CK (2005) GMAP: a genomic mapping and alignment program for mRNA and EST sequences. Bioinformatics 21 (9):1859–1875. https://doi.org/10.1093/bioinformatics/bti310
9. Gotoh O (2008) A space-efficient and accurate method for mapping and aligning cDNA sequences onto genomic sequence. Nucleic Acids Res 36(8):2630–2638. https://doi.org/10.1093/nar/gkn105
10. Gotoh O (2008) Direct mapping and alignment of protein sequences onto genomic sequence. Bioinformatics 24(21):2438–2444. https://doi.org/10.1093/bioinformatics/btn460
11. Li H (2018) Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics 34 (18):3094–3100. https://doi.org/10.1093/bioinformatics/bty191
12. Keller O, Odronitz F, Stanke M, Kollmar M, Waack S (2008) Scipio: using protein sequences to determine the precise exon/intron structures of genes and their orthologs in closely related species. BMC Bioinformatics 9:278. https://doi.org/10.1186/1471-2105-9-278
13. She R, Chu JS, Uyar B, Wang J, Wang K, Chen N (2011) genBlastG: using BLAST searches to build homologous gene models. Bioinformatics 27(15):2141–2143. https://doi.org/10.1093/bioinformatics/btr342
14. Iwata H, Gotoh O (2012) Benchmarking spliced alignment programs including Spaln2, an extended version of Spaln that incorporates additional species-specific features. Nucleic Acids Res 40(20):e161. https://doi.org/10.1093/nar/gks708
15. Csuros M, Holey JA, Rogozin IB (2007) In search of lost introns. Bioinformatics 23(13): i87–i96. https://doi.org/10.1093/bioinformatics/btm190
16. Csuros M (2008) Malin: maximum likelihood analysis of intron evolution in eukaryotes. Bioinformatics 24(13):1538–1539. https://doi.org/10.1093/bioinformatics/btn226

17. Gotoh O, Morita M, Nelson DR (2014) Assessment and refinement of eukaryotic gene structure prediction with gene-structure-aware multiple protein sequence alignment. BMC Bioinformatics 15:189. https://doi.org/10.1186/1471-2105-15-189
18. Gotoh O (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. J Mol Biol 264(4):823–838. https://doi.org/10.1006/jmbi.1996.0679
19. Gotoh O (1999) Multiple sequence alignment: algorithms and applications. Adv Biophys 36:159–206
20. Gotoh O (2000) Homology-based gene structure prediction: simplified matching algorithm using a translated codon (tron) and improved accuracy by allowing for long gaps. Bioinformatics 16(3):190–202. https://doi.org/10.1093/bioinformatics/16.3.190
21. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215(3):403–410. https://doi.org/10.1016/S0022-2836(05)80360-2
22. Gumbel EJ (1958) Statistics of extremes. Columbia University Press, New York
23. Dumas JP, Ninio J (1982) Efficient algorithms for folding and comparing nucleic acid sequences. Nucleic Acids Res 10(1):197–206. https://doi.org/10.1093/nar/10.1.197
24. Ma B, Tromp J, Li M (2002) PatternHunter: faster and more sensitive homology search. Bioinformatics 18(3):440–445. https://doi.org/10.1093/bioinformatics/18.3.440
25. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25(17):3389–3402. https://doi.org/10.1093/nar/25.17.3389
26. Eppstein D, Galil Z, Giancarlo R, Italiano GF (1992) Sparse dynamic-programming. 1. Linear cost-functions. J ACM 39(3):519–545. https://doi.org/10.1145/146637.146650
27. Gotoh O (2018) Modeling one thousand intron length distributions with fitild. Bioinformatics 34(19):3258–3264. https://doi.org/10.1093/bioinformatics/bty353
28. Slater GS, Birney E (2005) Automated generation of heuristics for biological sequence comparison. BMC Bioinformatics 6:31. https://doi.org/10.1186/1471-2105-6-31
29. Gotoh O (1990) Optimal sequence alignment allowing for long gaps. Bull Math Biol 52(3):359–373. https://doi.org/10.1007/bf02458577
30. Gotoh O (1982) An improved algorithm for matching biological sequences. J Mol Biol 162(3):705–708. https://doi.org/10.1016/0022-2836(82)90398-9
31. Iwata H, Gotoh O (2011) Comparative analysis of information contents relevant to recognition of introns in many species. BMC Genomics 12:45. https://doi.org/10.1186/1471-2164-12-45
32. Hirschberg DS (1975) Linear space algorithm for computing maximal common subsequences. Commun ACM 18(6):341–343. https://doi.org/10.1145/360825.360861
33. Myers EW, Miller W (1988) Optimal alignments in linear space. Comput Appl Biosci 4(1):11–17. https://doi.org/10.1093/bioinformatics/4.1.11
34. Hirschberg DS (1997) Serial computations of Levenshtein distances. Pattern matching algorithms. Oxford University Press, New York
35. Li W, Godzik A (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics 22(13):1658–1659. https://doi.org/10.1093/bioinformatics/btl158
36. Edgar RC (2010) Search and clustering orders of magnitude faster than BLAST. Bioinformatics 26(19):2460–2461. https://doi.org/10.1093/bioinformatics/btq461
37. Sedgewick R (1990) Algorithms in C. Addison-Wesley, Reading, MA
38. Wheeler TJ, Kececioglu JD (2007) Multiple alignment by aligning alignments. Bioinformatics 23(13):i559–i568. https://doi.org/10.1093/bioinformatics/btm226
39. Plyusnin I, Holm L (2012) Comprehensive comparison of graph based multiple protein sequence alignment strategies. BMC Bioinformatics 13:64. https://doi.org/10.1186/1471-2105-13-64
40. Gotoh O (1995) A weighting system and algorithm for aligning many phylogenetically related sequences. Comput Applic Biosci 11(5):543–551. https://doi.org/10.1093/bioinformatics/11.5.543
41. Gotoh O (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. Comput Appl Biosci 9(3):361–370. https://doi.org/10.1093/bioinformatics/9.3.361
42. Gotoh O (1994) Further improvement in methods of group-to-group sequence alignment with generalized profile operations. Comput Applic Biosci 10(4):379–387. https://doi.org/10.1093/bioinformatics/10.4.379

43. 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 Res 30 (14):3059–3066. https://doi.org/10.1093/nar/gkf436
44. Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32 (5):1792–1797. https://doi.org/10.1093/nar/gkh340
45. Gelfand MS, Mironov AA, Pevzner PA (1996) Gene recognition via spliced sequence alignment. Proc Natl Acad Sci U S A 93 (17):9061–9066. https://doi.org/10.1073/pnas.93.17.9061
46. Huang X, Zhang J (1996) Methods for comparing a DNA sequence with a protein sequence. Comput Applic Biosci 12 (6):497–506. https://doi.org/10.1093/bioinformatics/12.6.497
47. Mott R (1997) EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. Comput Applic Biosci 13 (4):477–478. https://doi.org/10.1093/bioinformatics/13.4.477
48. Usuka J, Brendel V (2000) Gene structure prediction by spliced alignment of genomic DNA with protein sequences: increased accuracy by differential splice site scoring. J Mol Biol 297 (5):1075–1085. https://doi.org/10.1006/jmbi.2000.3641
49. Zhang M, Gish W (2006) Improved spliced alignment from an information theoretic approach. Bioinformatics 22(1):13–20. https://doi.org/10.1093/bioinformatics/bti748

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

[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**

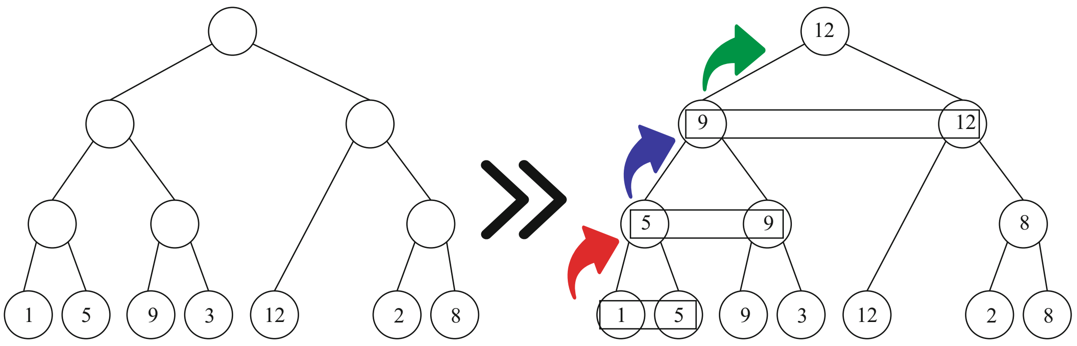


Fig. 1 From the naked guide tree, the algorithm starts from the leaf until the root labeling the internal nodes with the longest sequence of the children

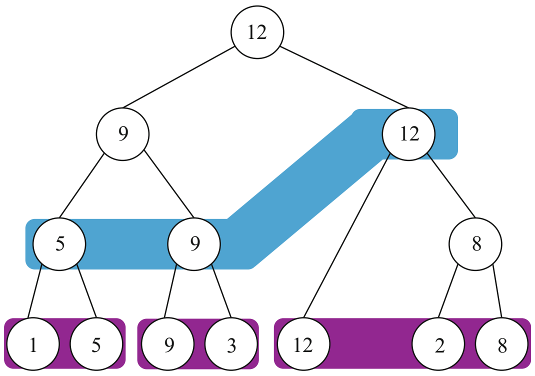


Fig. 2 From the root of the guide tree, *N* sequences are collected, ($N = 3$ in this example.) This is repeated recursively on the remaining nodes until all the leaf nodes are included in one of the subMSAs

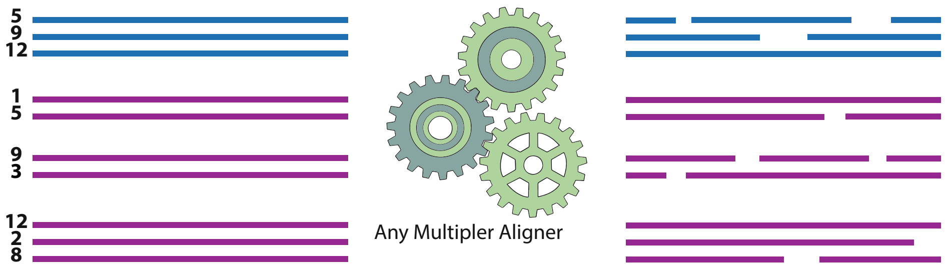


Fig. 3 Once all the small groups are defined, it is possible to generate the subMSAs in a parallel way using any third-party alignment software

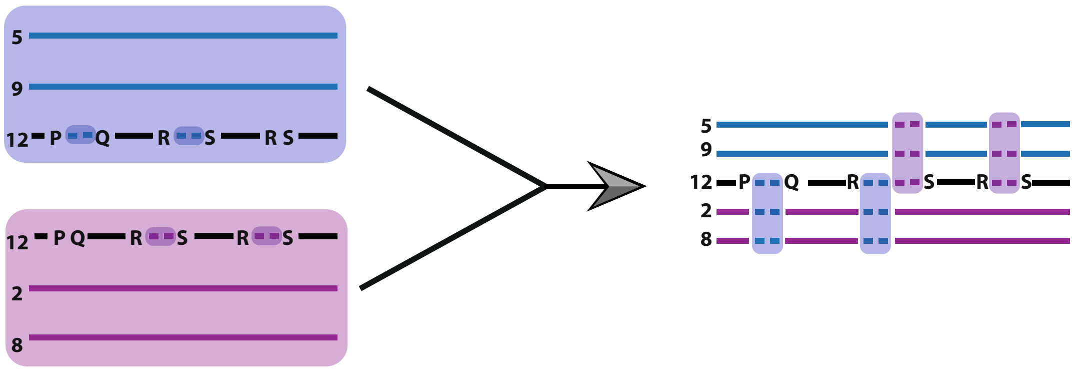


Fig. 4 Parent and children subMSAs are merged using the common sequence (12) projecting indels from parent to child and from child to parent

- 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
mafftest_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 7

Multiple Sequence Alignment for Large Heterogeneous Datasets Using SATé, PASTA, and UPP

Tandy Warnow and Siavash Mirarab

Abstract

The estimation of very large multiple sequence alignments is a challenging problem that requires special techniques in order to achieve high accuracy. Here we describe two software packages—PASTA and UPP—for constructing alignments on large and ultra-large datasets. Both methods have been able to produce highly accurate alignments on 1,000,000 sequences, and trees computed on these alignments are also highly accurate. PASTA provides the best tree accuracy when the input sequences are all full-length, but UPP provides improved accuracy compared to PASTA and other methods when the input contains a large number of fragmentary sequences. Both methods are available in open source form on GitHub.

Key words Multiple sequence alignment, PASTA, SATé, UPP, Ensembles of Hidden Markov Models

1 Introduction

Multiple sequence alignment (MSA) is one of the more complex bioinformatics tasks, and a precursor to many downstream analyses, including protein structure and function prediction, phylogeny estimation, orthology prediction, and even genome assembly. This chapter focuses mainly on the use of multiple sequence alignment for phylogeny estimation, and in particular on the challenge of computing alignments on large, heterogeneous datasets, where standard off-the-shelf methods have low accuracy. Of particular relevance is the challenge of computing multiple sequence alignments of datasets that exhibit sequence length heterogeneity, where all standard methods have particularly poor accuracy.

In 2009, SATé (Simultaneous Alignment and Tree Estimation) was developed to enable the co-estimation of alignments and trees on large challenging datasets [1]. SATé used a combination of divide-and-conquer (where alignments are computed on subsets using standard MSA methods and then merged into an alignment on the full dataset; *see* Fig. 1a) and iteration (where each iteration computes a new alignment based on the tree from the prior iteration, and then a new tree is computed on the new alignment) in order to obtain highly accurate alignments on large datasets. SATé-II was developed in 2012 [2] to improve on the accuracy and scalability of SATé; it used a modified decomposition strategy but otherwise had the same structure as SATé. SATé-II was able to run on much larger datasets than SATé, but was still limited to approximately 50,000 sequences. Finally, in 2014, the algorithmic design was changed again to produce PASTA [3, 4]. The objective in the design modification was to enable analyses of even larger datasets, but these changes also improved accuracy. Thus, PASTA, which mainly differs from SATé-II in its merging step (Fig. 1b), has the best accuracy and scalability of these three methods.

In 2015, we discovered that PASTA was unable to produce highly accurate alignments when the input dataset has many fragmentary sequences. To address this challenge, we developed UPP (Ultra-large alignments using Phylogeny-aware Profiles [5]), a new technique for alignment estimation that is based on a machine learning technique we developed, called an Ensemble of Profile Hidden Markov Models [6–8]. UPP uses PASTA to compute a "backbone alignment" of a subset of the input sequences (restricted to just the full-length sequences) and then adds the remaining sequences to the backbone alignment using a computed Ensemble of Profile Hidden Markov Models. UPP provides advantages over PASTA for datasets with fragmentary sequences, but PASTA has advantages over UPP when all the sequences are full-length. Like PASTA, UPP is able to compute highly accurate alignments on ultra-large datasets, including those with 1,000,000 sequences.

This chapter describes, at a very high level, how the PASTA and UPP algorithms operate, and provides some guidance on how to use these methods to obtain the best accuracy.[1] More information on how to run these methods can be obtained from the tutorials for PASTA and UPP available at the GitHub sites for these methods [10, 11].

2 SATé and PASTA

SATé [1], SATé-II [2], and PASTA [4] are methods for computing multiple sequence alignments and trees from unaligned sequences (*see* **Note 20**). They all have the same basic algorithmic strategy (Fig. 1), and so can be considered to be members of the same basic paradigm; however, SATé-II was designed to improve on SATé (now called SATé-I) and PASTA was designed to improve on

[1] This chapter is an update of [9], a previous article for Methods in Molecular Biology, which focused on using SATé [1, 2] for co-estimation of alignments and trees.

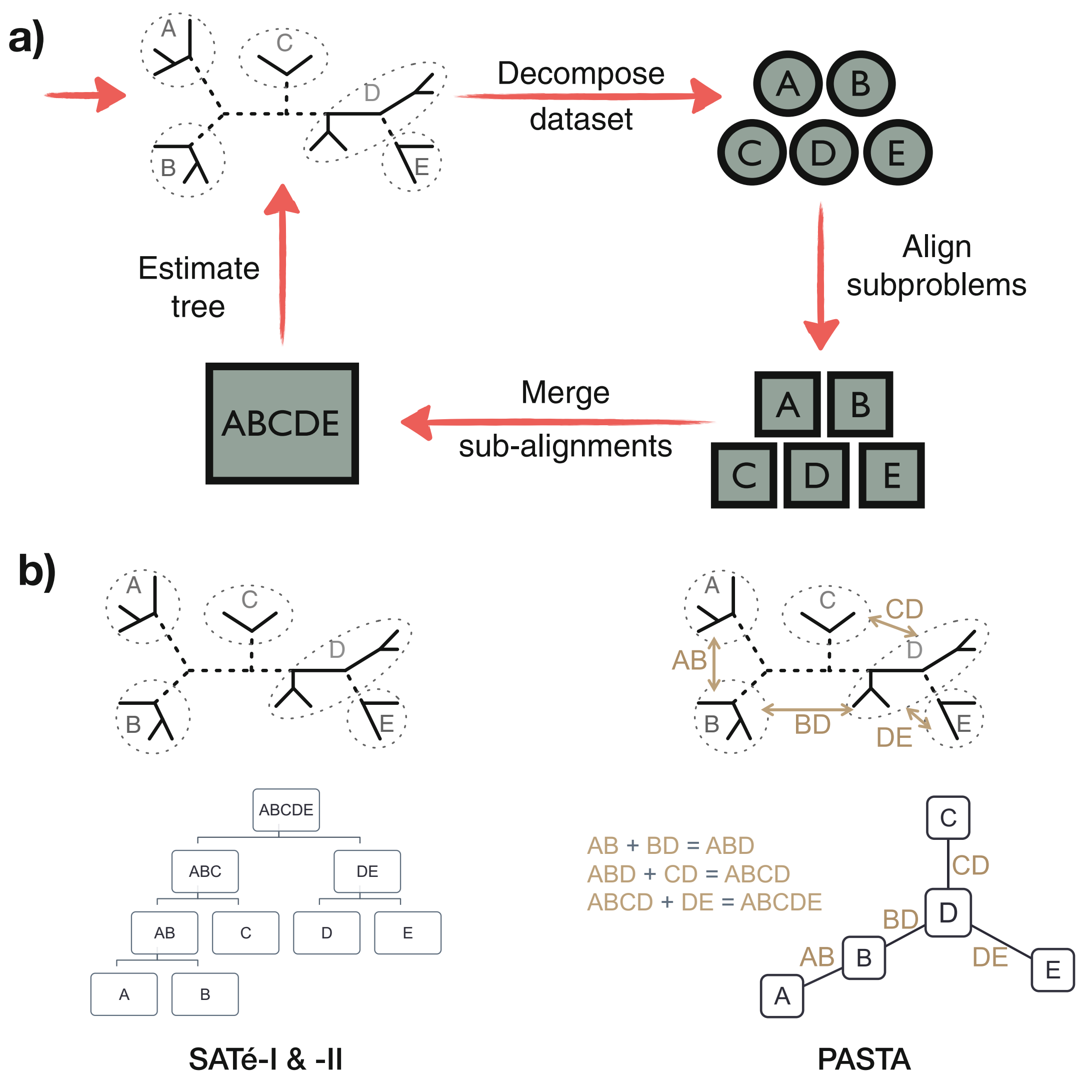


Fig. 1 (**a**) The general divide-and-conquer strategy used in SATé-I, SATé-II, and PASTA. In each iteration, using the current tree, sequences are divided into smaller subsets, each subset is aligned, alignments of subsets are merged, a new tree is inferred, and a new iteration starts. (**b**) SATé and PASTA differ mainly in how they merge sub-alignments. SATé uses a hierarchical approach, where the hierarchy reflects the tree, and uses external methods like Opal or Muscle to merge alignments. PASTA, on the other hand, uses a spanning tree, computed from the phylogeny, to compute a set of pairwise alignment mergers, which are then combined using transitivity

SATé-II, with the result that PASTA dominates the other methods with respect to accuracy, running time, memory usage, and scalability to large datasets. For example, PASTA has been able to compute alignments and trees on up to 1,000,000 sequences, but SATé-I and SATé-II have not been able to analyze datasets of this size. Furthermore, PASTA, which is the method of choice in this family of methods, has an active user community (e.g., Google group pasta-users@googlegroups.com) (*see* **Notes 1–3**).

2.1 Iterative Divide-and-Conquer Strategy

Each of these methods has the same basic structure. They begin by computing a quick alignment and tree, for example, using the fast maximum likelihood (ML) heuristic FastTree-2 [12] on a fast alignment, such as Clustal-Omega [13], and then they iterate between computing a new alignment using the current tree and computing an ML tree on the new alignment. The number of iterations (*see* **Note 10**) can be selected by the user or the user can simply run the method until some stopping criterion is met (e.g., the ML score stops improving). The final alignment/tree pair is then returned.

As noted, each iteration uses the tree from the previous iteration to compute a new alignment, and then a new ML tree is computed on the new alignment. The key to computing the new alignment is divide-and-conquer: the current tree is used to decompose the sequences into disjoint subsets, new alignments are computed on the subsets using a selected "subset aligner," and then the subset alignments are merged together into an alignment on the full dataset (Fig. 1a).

The only difference between SATé-I and SATé-II is that SATé-II enables the user to specify how large the subsets can be, and it modified the decomposition strategy so that the subsets do not exceed the specified maximum size; this change enables SATé-II to analyze larger datasets and results in improved accuracy compared to SATé-I. The major difference between SATé-II and PASTA is how the subset alignments are merged into a single alignment on the full dataset (Fig. 1b). The change in the sub-alignment merging strategy (in addition to other smaller changes, such as using a new method to obtain initial alignments) enables PASTA to analyze larger datasets than SATé-II and also improves its accuracy. In fact, PASTA can compute alignments on up to 1,000,000 sequences, and neither SATé-I nor SATé-II can analyze datasets of this size. Thus, PASTA strictly dominates SATé-I and SATé-II in terms of accuracy, scalability, and speed.

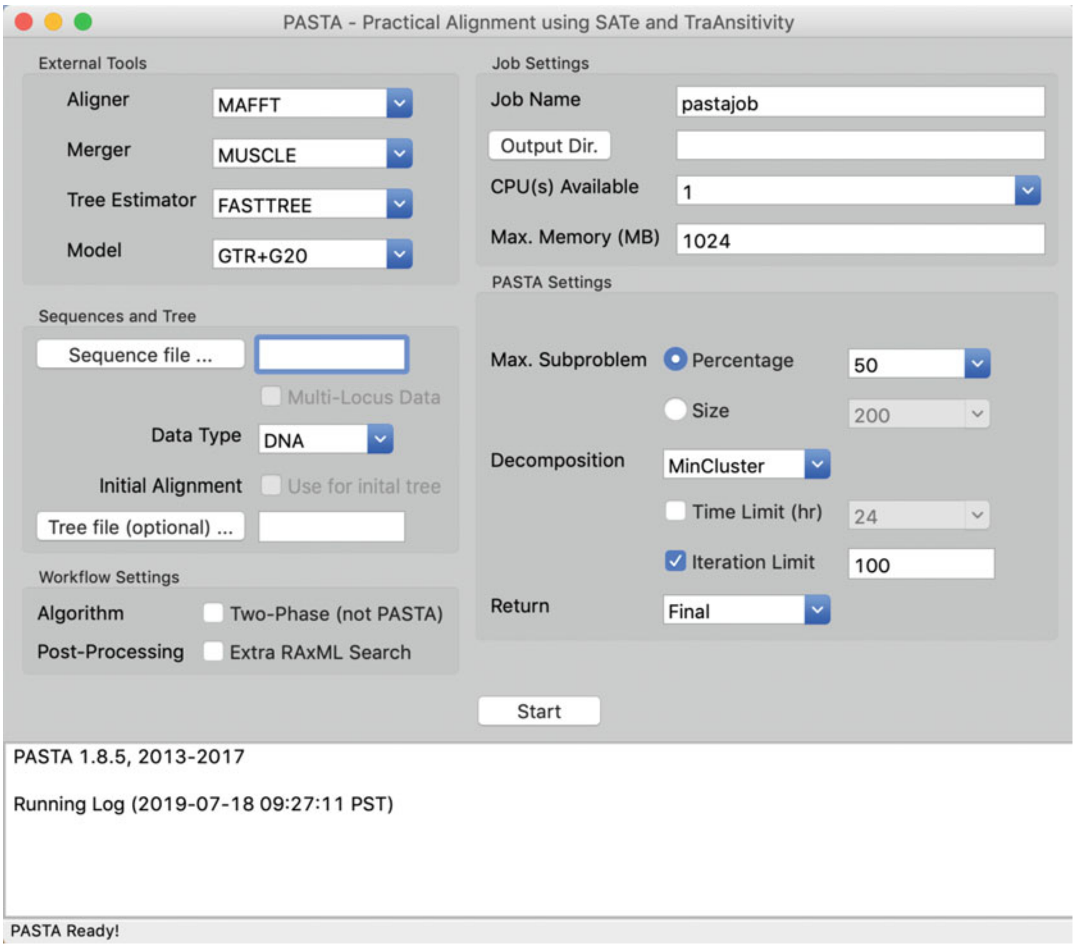
By design, PASTA is fundamentally a method for enabling a selected MSA method to be run only on subsets of bounded size (where the bound is selected by the user). Furthermore, PASTA provides many choices for the subset aligner, including MAFFT [14], Clustal-Omega, Opal [15], Prank [16], and Muscle [17], and additional subset aligner methods for protein sequences (*see* **Note 8**).

The rest of this section is described in terms of how to use the PASTA GUI (which is similar to the SATé GUI). However, the command line version of PASTA enables other options than the GUI, and so the advanced users should not restrict themselves to the GUI (*see* **Notes 1–3**).

2.2 PASTA Parameters

The PASTA GUI, shown in Fig. 2, shows the choices that the user has in running PASTA.

- **Aligner:** This option specifies the method used to compute alignments on subsets; the default is MAFFT, but other alignment methods are available. *See* **Notes 6–8**.
- **Merger:** This option allows the user to choose the method to merge pairs of alignments during its approach for combining


PASTA - Practical Alignment using SATe and TraAnsitivity

External Tools
Aligner: MAFFT
Merger: MUSCLE
Tree Estimator: FASTTREE
Model: GTR+G20

Sequences and Tree
Sequence file ...
Multi-Locus Data
Data Type: DNA
Initial Alignment: Use for inital tree
Tree file (optional) ...

Workflow Settings
Algorithm: Two-Phase (not PASTA)
Post-Processing: Extra RAxML Search

Job Settings
Job Name: pastajob
Output Dir.
CPU(s) Available: 1
Max. Memory (MB): 1024

PASTA Settings
Max. Subproblem: [✓ Percentage] 50; Size 200
Decomposition: MinCluster
Time Limit (hr) 24
[✓ Iteration Limit] 100
Return: Final

Start

PASTA 1.8.5, 2013-2017

Running Log (2019-07-18 09:27:11 PST)

PASTA Ready!

Fig. 2 PASTA graphical user interface (GUI). The GUI shows the major algorithmic choices in running PASTA; *see* Subheading 2.2. The EXTERNAL TOOLS determine how subsets are aligned, how these subset alignments are merged, and how trees are computed on the merged alignment in each iteration (which is determined both by the tree estimator method and the sequence evolution model). SEQUENCES AND TREE indicate the type of data, and also allow the user to provide an initial alignment and tree. The PASTA SETTINGS specify the maximum size for the subsets, the type of decomposition used in decomposing the dataset into subsets, how many iterations to perform, and whether to return the tree from the last iteration or the tree (from among all the iterations) with the best maximum likelihood score. Finally, WORKFLOW SETTINGS allow the user to just perform a two-phase analysis (first compute an alignment and then a tree) instead of using PASTA, or to run RAxML [18] on the final alignment returned by PASTA

subset alignments into an alignment on the full dataset; the choice is between Opal and Muscle. The merger technique in PASTA is only used on *pairs* of alignments, and the final alignment is then constructed from these merged pairs using transitivity. *See* **Note 9**.

- **Tree Estimator:** This option allows the user to choose between RAxML and FastTree-2, two heuristics for maximum likelihood, for computing trees in each iteration. The default is FastTree-2; *see* **Note 11**.
- **Model:** This option specifies the sequence evolution model, but this depends on the data type (RNA, DNA, or protein) as well as the tree estimator (RAxML or FastTree-2); *see* **Notes 12–14**.
- **Data Type:** This section allows the user to specify the data type (DNA, RNA, or protein). The default for the data type is DNA, and unless the user specifies otherwise, the analysis will be performed as though the data are DNA. *See* **Note 6**.
- **Initial alignment:** This is an optional command that allows the user to provide a pre-computed alignment to PASTA, for use in computing the first tree. *See* **Note 4**.
- **Tree file:** This is an optional command that allows the user to provide a pre-computed tree to PASTA, for use in computing the first decomposition and subsequent alignment. *See* **Note 4**.
- **Max. Subproblem:** The options here let the user specify the maximum subset size, either as a percentage of the full set of sequences or as a fixed number (i.e., "size"). *See* **Notes 5**, **7**, and **15**.
- **Decomposition:** This option specifies both which edges to remove (MinCluster, centroid edge or longest edge) in computing the decomposition of the sequences into subsets (the default is MinCluster) and how many iterations to perform (either a specific number of iterations or a maximum amount of time). The MinCluster decomposition, which minimizes the number of subsets with a bounded size [19], began with version 1.8.0, and it further improves alignment accuracy.
- **Return:** This option allows the user to decide whether to return the tree from the last iteration or the tree with the best maximum likelihood score of all the trees from all the iterations. *See* **Note 16**.

The most important considerations in running PASTA are (1) what alignment method to use to compute alignments on subsets, (2) how small to make the subsets, and (3) how many iterations to run. A good default for the subset aligner is MAFFT [14]. When MAFFT is used to align subsets, then limiting the subsets to 200 sequences makes it feasible to run the more computationally intensive variants of MAFFT (such as MAFFT L-INS-i and MAFFT G-INS-i), which improves accuracy; reducing the maximum subset size will tend to reduce the running time while increasing the maximum subset size will tend to increase the running time (*see* **Note 7** for a discussion about the impact on accuracy). Other methods, such as BAli-Phy [20, 21], can also be used to align subsets, as shown in [22].

The number of iterations is also important, and previous studies have shown that accuracy improves substantially in the first few iterations, and then alignment and tree accuracy seem to stabilize. While three iterations seem to be sufficient for high accuracy in most conditions, it seems possible that more iterations could improve accuracy for challenging datasets. However, increasing the number of iterations also increases the running time. Hence, this is an issue that involves a potential tradeoff between time and accuracy; *see* **Note 4**.

The tree estimation method used in PASTA within each iteration also impacts accuracy, and the default is FastTree-2. Most users will prefer to use other methods than FastTree-2 for the final tree, and PASTA enables the user to perform a final RAxML analysis on the final tree. This is advisable whenever the dataset is not so large that RAxML is infeasible. *See* **Notes 11** and **14**.

2.3 PASTA Output

PASTA produces both an alignment and a tree. In addition to the final alignment and tree, PASTA outputs alignments and trees generated in each iteration as temporary files. It also outputs a config file recording all the settings used.

For large datasets (many thousands of sequences), PASTA tends to produce very long and gappy alignments because it is conservative in inferring homologies. The gappy alignments (partially a natural consequence of large datasets and partially a consequence of the algorithmic design of PASTA) seem to not hurt PASTA's ability to produce very accurate trees, but the alignments will certainly look strange to some users (see discussion in [23] about the preference among some users for less gappy alignments). Furthermore, whether accurate or not, these gappy alignments can also cause difficulties in some subsequent analyses. For example, phylogenetic inference can become slow given long alignments, and the inclusion of gappy sites may not result in improved phylogenetic accuracy. To speed up the tree estimation stage within each iteration, PASTA alignments are first masked to remove all sites that are at least 99.9% gapped, before trees are computed. This default setting for masking sites inside PASTA can be modified using the `--mask-gappy-sites` option. Removing gappy sites from the final alignment generated by PASTA can be done using the `run_seqtools.py` script, which is packaged with and installed together with PASTA. More aggressive filtering would further reduce the running time for phylogenetic inference, but could also have negative consequences for accuracy; see discussion in [24].

2.4 Websites for PASTA

- PASTA is available in open source form on GitHub at [10] and a protein version is available at [25]. The software is developed under the GNU Public License (GPL).
- A version of PASTA for use with BAli-Phy is available at [26].
- All questions and inquires should be addressed to our user email group: https://pasta-users@googlegroups.com, with posts available at https://groups.google.com/forum/#!forum/pasta-users.
- A PASTA tutorial is available at https://github.com/smirarab/pasta/blob/master/pasta-doc/pasta-tutorial.md.

3 UPP

As we have found, PASTA produces highly accurate alignments and trees, and improves on the accuracy of other methods for ultra-large datasets with high rates of heterogeneity. However, when the input dataset has many fragmentary sequences, then PASTA does not have good accuracy. Furthermore, no standard alignment method has good accuracy when fragments are included. However, UPP [5] is an alternative approach that has good accuracy, and is the focus of this section.

3.1 Ensembles of Profile Hidden Markov Models (HMMs)

UPP builds on PASTA to improve its ability to align datasets with fragmentary sequences using the "ensembles of HMMs" technique, which we now describe in the context of working with multiple sequence alignments.

A profile Hidden Markov Model (HMM) [27] is a probabilistic graphical model that has vertices and directed edges, with a single vertex for the start state, a single vertex for the end state, and additional vertices corresponding to match states, insertion states, and deletion states. With the exception of the insertion states (which can have self-loops), there are no directed cycles in a profile HMM. Each directed edge $e = v \rightarrow w$ in the profile HMM is annotated with a real number p_e where p_e is the probability of moving from v to w. Finally, the insertion states and match states emit letters (e.g., nucleotides or amino acids) from a probability distribution. Thus, when tracing a path through a profile HMM, and selecting the letters to be emitted by the visited match and insertion states, a sequence is produced.

Profile HMMs are a major part of many bioinformatics analyses, and one of the interesting uses is to add sequences into multiple sequence alignments. In what follows, we describe how profile Hidden Markov Models can be used specifically for multiple sequence alignment; *see* [28] for additional details and discussion.

To add a sequence *s* into a multiple sequence alignment *A*, a profile HMM is built for *A*, and then an optimal path (e.g., a maximum likelihood path) through the model is found for *s*. Once the path is found, it defines a way of adding *s* into the alignment *A*. Note that this addition does *not* define an alignment between *A* and those letters in *s* that are mapped to insertion states. Thus, when using HMMs to extend *A* to include *s*, some parts of *s* may remain *unaligned*. Besides finding the best alignment, given the sequence *s* and a profile HMM, the fit between the profile HMM and *s* can be calculated in various ways, including finding the overall probability that the profile HMM would generate *s*. The HMMER3 [29] suite of tools provides a particular implementation of the general profile HMM concept and includes many further optimizations, both for accuracy and speed. HMMER3 includes tools for all these analyses (i.e., building profile HMMs from alignments, scoring the fit between a profile HMM and a sequence, and finding the best path through the model for the sequence) [29, 30].

An *ensemble of profile HMMs* is a collection of profile HMMs that are built using a multiple sequence alignment *A*, with each profile HMM in the set based on just a subset of the sequences in the set. Thus, the match states in each of the profile HMMs in the set correspond to sites in *A*. Now, given a sequence *s*, the profile HMM in the collection that has the best fit to *s* can be found, the best path through the model can be computed, and thus the sequence *s* can be added to the alignment *A*. Thus, an ensemble of profile HMMs can also be used to represent the alignment *A* and then used to add new sequences to *A*. Here we will show how UPP uses an ensemble of profile HMMs to compute multiple sequence alignments, noting also that ensembles of HMMs have been used for phylogenetic placement [6], taxonomic identification of metagenomic data [7], and classification of protein sequences into families and superfamilies [8].

3.2 UPP's Algorithmic Protocol

In essence, UPP is a combination of PASTA (which it uses to construct a multiple sequence alignment on a subset of the input sequences) with a way of computing an ensemble of profile HMMs, which it then uses to add the remaining sequences into the PASTA alignment. Here we describe this process as operating in four steps. The first two steps can be omitted if the user wishes to provide UPP with a pre-computed backbone alignment and tree; *see* **Note 17**.

1. Given a set *S* of unaligned sequences, UPP begins by identifying those sequences to be part of "backbone alignment." This is performed first by restricting *S* to just those sequences with length within 25% of the median sequence length, and then randomly selecting a set of sequences from that set. The number of sequences in the backbone and restrictions on what

sequences can be included in the backbone can be modified by using a configuration file or input options (`-B`, `-M`, `-T`, and `-l`).

2. UPP uses PASTA to compute a multiple sequence alignment A' and tree T on S', which are then referred to as the backbone alignment and backbone tree.
3. UPP builds an ensemble of profile HMMs to represent the multiple sequence alignment A' on S': it uses the tree T to break the set of sequences into disjoint subsets of bounded size (using the same centroid edge decomposition as in SATé-II), and then computes a profile HMM for each of the subsets (i.e., for the rows of the alignment A' defined by the sequences in the subset). The set of profile HMMs it creates is the ensemble of profile HMMs used in the next step.
4. The remaining sequences (i.e., the ones that are not in the backbone alignment) are added to A' using the ensemble of profile Hidden Markov Models computed in **Step 3**, thus producing a multiple sequence alignment A on S.

As shown in [5], UPP produces more accurate alignments than PASTA and other multiple sequence alignment methods when the input set S has many fragmentary sequences, and trees computed on the alignment are more accurate than trees computed on the other alignments in the presence of fragmentation.

3.3 UPP's Parameters

The most important algorithmic options in using UPP are (a) which sequences to put in the backbone subset S', (b) which method to use to compute an alignment on S', and (c) which algorithmic parameters to use for building the ensemble of profile Hidden Markov Models.

For which sequences to put in S', there are two decisions that need to be made: first, which sequences are close enough to full-length to be considered, and second, how many of these sequences to use for the backbone alignment. The default UPP operates as follows: it computes the median sequence length of the input sequences and considers any sequence within 25% of this length to be "full-length." Then, UPP selects a random subset of the "full-length" sequences to include in the backbone alignment, with the default setting for the size of this set being the minimum of $\{1000, N\}$, where N is the number of "full-length" sequences. Changing the number of sequences to put in the backbone set can affect accuracy and running time, and is discussed in *see* **Note 18**.

For how to compute the backbone alignment, the default is to use PASTA, and this is certainly appropriate when S' is large. However, when S' is small enough, then other methods can potentially provide improved accuracy compared to PASTA. For example, BAli-Phy [20, 21] and other statistical methods could be used to compute an alignment A' on S'. Once the backbone alignment is built, a backbone tree is also needed, which can be estimated using fast ML heuristics, such as FastTree-2 (e.g., as outputted by PASTA).

There are several algorithmic options for building the ensemble of profile HMMs on A', which we briefly discuss here. Recall that an ensemble of profile HMMs is a collection of profile HMMs, where each of the profile HMMs is constructed on a subset of the sequences in the backbone alignment. To add a sequence s into the backbone alignment, s is scored with respect to each profile HMM in the collection, and the profile HMM with the best score is selected. Thus, every sequence s that is not in the backbone alignment must be scored against every profile HMM in the collection. Although we observe that typically accuracy is increased by having a large number of profile HMMs, this also increases the running time. Thus, there is a potential tradeoff between accuracy and running time. The default in UPP produces 10 profile HMMs, which provides an improvement over a single profile HMM and (obviously) also increases the running time. *See* **Note 19** for additional considerations for this algorithmic setting.

Although modifications to the default settings can result in improved accuracy or speed, the default settings for UPP are sufficient to improve on PASTA if the proportion of fragmentary sequences is large enough. Detailed information on how to adjust the settings of UPP are given in its README file at https://github.com/smirarab/sepp/blob/master/README.UPP.md.

3.4 Websites for UPP

- The UPP software is available in open source form on GitHub at [11], and is part of the SEPP [6] distribution (which has code for various methods that use ensembles of profile HMMs). UPP is available as Python code.
- A tutorial on UPP is available at https://github.com/smirarab/sepp/blob/master/tutorial/upp-tutorial.md
- The UPP users group forum is available at https://groups.google.com/forum/#!forum/ensemble-of-hmms.

4 Discussion and Summary

UPP and PASTA are two methods for large-scale multiple sequence alignment that provide improved accuracy over standard methods when datasets are large and heterogeneous. UPP provides a specific advantage over PASTA when the dataset has fragmentary sequences and PASTA provides advantages when all the sequences are full-length. UPP and PASTA are available in open source form in order to encourage further development by the research community. Furthermore, each method is designed to improve scalability of

MSA methods, which are run only on subsets of the input sequence set. Therefore, as new MSA methods are developed, PASTA and UPP can be extended to use these new methods.

PASTA is described here as a method for co-estimating alignments and trees, but it is not a statistical co-estimation method in the sense that BAli-Phy [20, 21] and StatAlign [31] are. However, PASTA can run on very large datasets while truly statistical co-estimation methods are limited to fairly small datasets (perhaps 100 sequences). Furthermore, PASTA and UPP have been used with BAli-Phy to compute subset alignments [22], thus enabling BAli-Phy to scale (in some sense) to very large datasets (e.g., up to 10,000 sequences!).

Although the discussion here was largely based on using PASTA within the GUI, the command line version enables additional settings that can provide improved accuracy. This was intentional, as the GUI is the easiest way to become familiar with PASTA, and the GUI version provides the same advantages as the command line version over other methods on large datasets. However, advanced users should use the command line version, which allows the algorithmic settings to be modified in additional ways.

We set out to discuss multiple sequence alignment for the purpose of tree estimation. PASTA is specifically designed to co-estimate alignments and trees (in an iterative fashion), so that the final tree is produced by running a maximum likelihood heuristic (either RAxML or FastTree-2) on the final alignment. Some consideration, therefore, should be made for how to compute trees from these improved alignments. While we focused on maximum likelihood under standard sequence evolution models, other approaches could be used, including Bayesian estimation (e.g., MrBayes [32] and BEAST [33, 34]), distance-based estimation (e.g., FastME [35]), and parsimony analyses (e.g., TNT [36] and PAUP* [37]). Bayesian or maximum likelihood analyses under non-standard sequence evolution models may also be necessary, especially for datasets that span large evolutionary distances where violations of the usual model assumptions (stationarity, time reversibility, and homogeneity) are likely to occur [38–40]. Divide-and-conquer phylogeny estimation, where the set of species is divided into smaller, more homogeneous subsets, and then trees on the subsets are computed and combined into a tree on the full dataset (e.g., DACTAL [41], constrained-INC [42, 43], NJMerge [44], and TreeMerge [45]), may provide an improvement in tree accuracy for those datasets that violate the standard model assumptions but are too large for methods that are based on more complex models.

Finally, although UPP was able to produce better alignments than PASTA for datasets with a high number of fragmentary sequences, the construction of trees from such datasets presents additional challenges, even given error-free alignments [46]. One possible direction is to use phylogenetic placement, where an initial tree is built using the full-length sequences and then the fragmentary sequences are added to the tree [6], but other approaches may provide better accuracy. Thus, tree estimation on large heterogeneous datasets will need to be revisited, in order to achieve the goal of accurate inference of large phylogenies.

5 Notes

We now give some high-level advice on using PASTA and UPP. The first 16 notes are for PASTA, the next three notes are for UPP, and the final note is common to both methods. The reader will benefit from consulting the GitHub sites for these methods (and in particular the tutorials and READMEs at those sites). PASTA users should also read the Notes section in [9] for advice about using PASTA (which is built on the SATé codebase, so that much of the advice for SATé is relevant to PASTA).

1. If you have a MAC, then installing PASTA by downloading the MAC application .dmg from the GitHub site is easy, but it only allows you to use the GUI (which is not always the most up-to-date version of PASTA). If you prefer to use the command line or do not have a MAC, you will need to install PASTA using some other process. The PASTA GitHub site provides details on how to do these installations, and the PASTA users group can help with installation issues.
2. PASTA has been mainly developed and tested for Linux and MAC; as a result, Windows users will generally have more difficulty and will need to rely on virtualization (through virtual images or docker images provided on the website).
3. PASTA utilizes FASTA-formatted sequence files and Newick-formatted tree files. See the PASTA README for details about allowed characters in the input data.
4. PASTA uses iteration as well as divide-and-conquer to improve alignment accuracy compared to standard MSA methods. The main algorithmic parameters (i.e., how small to make the subsets, how to compute subset alignments, how to compute trees on the alignments and which sequence evolution models to use) impact the accuracy that can be obtained in each iteration, but also impact running time. In general, our recommendation is to use the best method you can afford to run that still allows PASTA to perform at least three iterations (and more iterations, when time permits). This will allow the alignment produced by PASTA to have very good accuracy, and a final tree can then be computed on the PASTA alignment using more computationally intensive tree estimation methods. Much of

the discussion below about how to set the algorithmic parameters reflects this point. Similarly, if desired, the final alignment/tree pair produced by PASTA can be given as input to PASTA (see Sequences and Tree in the PASTA GUI, in Fig. 2), if additional PASTA iterations using more computationally intensive approaches are desired.

5. PASTA and SATé were designed to enable improved accuracy on large datasets, but they have also been used to compute alignments on small datasets (e.g., the avian datasets in [47] with fewer than 50 taxa). The PASTA default setting automatically adjusts the subset size appropriately for small datasets. For example, on sufficiently small datasets, PASTA may set the maximum subset size to as much as 50% of the number of sequences in the input.
6. PASTA (in command line mode) does not automatically detect the data type (DNA, RNA, or proteins), and the default setting is DNA. Therefore, if your data are not DNA sequences, you should make sure to specify the type explicitly, as otherwise the behavior of PASTA can be unpredictable (and the resultant alignment and tree may have poor accuracy).
7. As mentioned above, MAFFT is the default technique for aligning subsets, and works well for both proteins and nucleotides. However, when aligning proteins, other subset alignment methods can also have good accuracy, and are enabled in [25]. As mentioned earlier, when MAFFT is used to align subsets, limiting the subsets to 200 sequences makes it feasible to run the most accurate (but also most computationally intensive) variants of MAFFT, such as MAFFT L-INS-i and MAFFT G-INS-i. Changing the subset size will change the final alignment. Our studies (published and unpublished) have revealed inconsistent trends regarding the impact of the subset size parameter. However, at this time, based on the preponderance of the evidence, we suggest using the default settings, which puts the alignment subset size at 200, when using MAFFT as the subset aligner. The interested user may wish to explore the impact of changing alignment subset size, for those datasets that are small enough to allow such exploratory data analysis.
8. For protein alignment, PASTA enables the use of additional subset aligners MAFFT-G-INS-i, MAFFT-homologs, CONTRAlign (version 1) [48], and PROBCONS [49, 50]. To use MAFFT-homologs and CONTRAlign (available only in command line), the user must take additional steps during installation, as detailed in the most up to data README file. If you wish to use MAFFT-Homologs as the subset aligner, you should use the version of PASTA available at [25].

9. PASTA allows two methods for merging pairs of alignments—Opal and Muscle. The choice between the two methods does not have a large impact on accuracy, provided that the subsets are not too small (because when the subsets are very small, then the returned alignment is largely based on the technique used to merge pairs of alignments).

10. Although the default setting for PASTA sets the number of iterations to three, additional iterations could lead to improved accuracy under some conditions. In general, using additional iterations has shown some improvement in tree and alignment accuracy, but the optimal number of iterations is an under-explored topic. We therefore recommend that the user consider enabling additional iterations, when time permits, and explore the set of alignment/tree pairs that are returned in these iterations.

11. PASTA has two methods for tree estimation that are used in each iteration. The default is FastTree-2, but RAxML is also allowed. In our experience, RAxML is much more computationally intensive than FastTree-2, making FastTree-2 a better choice on large datasets, since many iterations can be run if FastTree-2 is used instead of RAxML (see previous Note). When the number of sequences is small enough, then adding a final RAxML run (with the post-processing command in the GUI) is recommended, since RAxML generally produces better ML scores and can, in some conditions, improve the tree accuracy (although there are many conditions where the improvement in ML score does not correspond to an improvement in tree topology accuracy [51]). However, when the number of sequences is large then we do not recommend having PASTA automatically perform a RAxML analysis on the final alignment, as this can be too computationally intensive. Instead, for very large datasets, we recommend the following approach: let PASTA perform its iterations using FastTree-2, save the final PASTA alignment, and then separately compute a tree on the final PASTA alignment using the preferred software (e.g., RAxML, or potentially some other method) and selected sequence evolution model. In this way, PASTA can be used to produce a highly accurate alignment, and then the best tree accuracy (and associated numeric parameters) can be obtained using a separate tree estimation phase.

12. The set of possible sequence evolution models depends on the tree estimation method (RAxML or FastTree-2) and the type of data (nucleotides or proteins). When PASTA is used with FastTree-2, only a very limited number of models are available (described in the notes below). If the user wishes to select a model for PASTA, then they should obtain a preliminary alignment and then use external software (e.g., ProtTest [52] for

protein datasets and ModelTest [53] or PLTB [54] for nucleotide datasets). However, an alternative approach can also be used: the user can run PASTA using the default model, then use the resultant alignment with the external software to select a substitution model for a final round of tree inference or potentially another iteration of PASTA (using the new model). See discussion in [28] about selecting models for phylogeny estimation.

13. To select a nucleotide sequence evolution model within PASTA, FastTree-2 and RAxML both enable the Generalized Time Reversible (GTR, [55]) model, and each can be used with a selected model for rate variation across sites (with different models depending on what tree estimation method is selected). In addition, FastTree-2 enables the use of the Jukes–Cantor (JC, [56]) model (with two models for rate variation across sites); however, we do not recommend using the JC model unless the data seem to fit the JC model well. FastTree-2 only enables two types of rate variation across sites (CAT and G20, which is an approximation of gamma distributed rates with 20 categories), but RAxML enables rate variation models that include invariable sites. The choice of rate variation model can also impact accuracy, but the more complex models are also more computationally intensive. However, our studies suggest that using simple sequence evolution models within the iterative process may not reduce the alignment accuracy substantially, and a new tree can be estimated on the final alignment using more complex models.

14. For protein alignment, the two tree estimation methods, RAxML and FastTree-2, offer very different sequence evolution models. Specifically, FastTree-2 only offers two protein substitution models (JTT and WAG), each with two site variation models, and RAxML offers 11 protein substitution models, each with four site variation models. Thus, RAxML allows a larger set of protein sequence evolution models than FastTree-2, making RAxML a better method for computing trees than FastTree-2 for proteins. However, here too the benefit from using RAxML within the iterative process may be offset by the extra time used to compute trees with RAxML. Hence, we would suggest instead that FastTree-2 be used as the tree estimation method within the iterative procedure, even for protein sequences. Then, after the PASTA alignment is computed, the user can compute a new tree on the alignment using RAxML or some other software, under the best fitting model.

15. If you wish to use BAli-Phy to align subsets within PASTA, the maximum subset size and the running time for each subset alignment need to be set so that BAli-Phy is able to converge on each subset. This is discussed in [57], and the

software for PASTA using BAli-Phy is available at [26]. However, *see* [58] for a study comparing BAli-Phy and other alignment methods on protein benchmark datasets, which showed differences between performance on simulated and biological datasets.

16. The choice of which alignment/tree pair to return (i.e., whether to return the pair produced in the final iteration or the pair that has the best maximum likelihood score) is an interesting one. In general, we expect little difference in accuracy between the two options, and so the choice may not make much difference in practice. In addition, there is no theoretical basis on which to select the pair that has the best maximum likelihood score [2], since the alignment is allowed to change. For these reasons, and also because the ML score is calculated within PASTA on masked versions of the computed alignments, the default in PASTA is the final alignment/tree pair.
17. The user can provide UPP with a pre-computed backbone alignment and tree (referred to in the UPP tutorial as a "custom seed alignment and tree"); this is a natural approach when using alignments and trees obtained from external sources (such as PFAM [59]) or when alignments and trees have been estimated using additional information (such as secondary or tertiary structure) or by specialized methods not available within UPP.
18. UPP uses PASTA to compute its backbone alignment and tree, but the selection of which sequences are put into the backbone set can be controlled by the user. In the default mode, UPP operates as follows: it computes the median sequence length of the input sequences and considers any sequence within 25% of this length to be "full-length"; the user can modify this approach as needed using options `-M` and `-T`. Once that set of full-length sequences is determined, the user can specify how many of the sequences to include in the backbone alignment using the `-B` option. The default is to take the minimum of {1000, N}, where N is the number of "full-length" sequences. However, another option is to include all of the full-length sequences (even when this is more than 1000); in our experience, this improves accuracy but may also increase the running time. Furthermore, reducing the number of sequences, even to as low as just 100 (the UPP-fast version), produces a reduction in accuracy (but sometimes only a small reduction, which depends on the heterogeneity in the input dataset) and a dramatic reduction in running time. Hence, there is a potential tradeoff between accuracy and running time that needs to be considered in building the ensemble.

19. The default mode for UPP is to create an ensemble of HMMs that has ten (10) profile HMMs. However, changes to this number can be considered with a potential for improved accuracy. In particular, when the input set is highly heterogeneous (as represented by low average sequence similarity), then using a larger number of profile HMMs can improve accuracy; however, the benefit in increasing the number of profile HMMs is reduced when the dataset has high average sequence similarity. Furthermore, increasing the number of profile HMMs automatically increases the running time (as it scales linearly with this number).

20. Errors in the input unaligned sequence data have the potential to reduce the accuracy of the alignment. One way to detect such errors is to use automated methods such as TreeShrink [60]. TreeShrink looks for extremely long branches in the phylogeny to detect potential errors in the data. Thus, TreeShrink can be combined with PASTA in a natural way: remove sequences on long branches from the PASTA alignment and tree (implemented in the `--treeshrink-filter` option), recompute the PASTA alignment, and add back those potentially problematic sequences using UPP. In addition, UPP allows for sequences that are on very long branches to be removed from the backbone set (see `-l`).

Acknowledgements

This paper was supported by NSF grant ABI-1458652 to TW and NSF grant IIS-1845967 to SM.

References

1. Liu K, Raghavan S, Nelesen S, Linder CR, Warnow T (2009) Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science 324 (5934):1561–1564
2. Liu K, Warnow T, Holder MT, Nelesen SM, Yu J, Stamatakis AP, Linder CR (2012) SATé-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. Syst Biol 61(1):90–106
3. Mirarab S, Nguyen N, Warnow T (2014) PASTA: ultra-large multiple sequence alignment. In: International conference on research in computational molecular biology. Springer, Berlin, pp 177–191
4. Mirarab S, Nguyen N, Wang L-S, Guo S, Kim J, Warnow T (2015) PASTA: ultra-large multiple sequence alignment of nucleotide and amino acid sequences. J Comput Biol 22:377–386
5. Nguyen N, Mirarab S, Kumar K, Warnow T (2015) Ultra-large alignments using phylogeny aware profiles. Genome Biol 16:124. A preliminary version appeared in the Proceedings RECOMB 2015
6. Mirarab S, Nguyen N, Warnow T (2012) SEPP: SATé-enabled phylogenetic placement. In: Pacific symposium on biocomputing, pp 247–58
7. Nguyen N, Mirarab S, Liu B, Pop M, Warnow T (2014) TIPP: taxonomic identification and phylogenetic profiling Bioinformatics 30 (24):3548–3555
8. Nguyen N, Nute M, Mirarab S, Warnow T (2016) HIPPI: highly accurate protein family classification with ensembles of hidden Markov

models. BMC Bioinformatics 17(Suppl 10):765

9. Liu K, Warnow T (2014) Large-scale multiple sequence alignment and tree estimation using SATé. In: Multiple sequence alignment methods. Springer, Berlin, pp 219–244
10. Mirarab S (2019) Github site for PASTA software. https://github.com/smirarab/pasta. Accessed 13 July 2019
11. Mirarab S (2019) Github site for Ensemble of HMM methods (SEPP, TIPP, UPP) software. https://github.com/smirarab/sepp. Accessed 13 July 2019
12. Price MN, Dehal PS, Arkin AP (2010) FastTree 2 – approximately maximum-likelihood trees for large alignments. PLoS ONE 5(3), e9490. https://doi.org/10.1371/journal.pone.0009490
13. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Soding J, Thompson JD, Higgins DG (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7:539
14. Katoh K, Toh H (2008) Recent developments in the MAFFT multiple sequence alignment program. Brief Bioinf 9(4):286–298
15. Wheeler T, Kececioglu J (2007) Multiple alignment by aligning alignments. In: Proceedings of the 15th ISCB conference on intelligent systems for molecular biology, pp 559–568
16. Löytynoja A, Goldman N (2005) An algorithm for progressive multiple alignment of sequences with insertions. Proc Nat Acad Sci 102:10557–10562
17. Edgar RC (2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 5(113):113
18. Stamatakis A (2006) RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models Bioinformatics 22:2688–2690.
19. Balaban M, Moshiri N, Mai U, Mirarab S (2019) TreeCluster: clustering biological sequences using phylogenetic trees. bioRxiv, https://doi.org/10.1101/591388
20. Suchard MA, Redelings BD (2006) BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22:2047–2048
21. Redelings BD, Suchard MA (2007) Incorporating indel information into phylogeny estimation for rapidly emerging pathogens. BMC Evol Biol 7:40
22. Nute M, Warnow T (2016) Scaling statistical multiple sequence alignment to large datasets. BMC Genomics 17(10):764
23. Löytynoja A, Goldman N (2008) Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. Science 320(5883):1632–1635
24. Tan G, Muffato M, Ledergerber C, Herrero J, Goldman N, Gil M, Dessimoz C (2015) Current methods for automated filtering of multiple sequence alignments frequently worsen single-gene phylogenetic inference. Syst Biol 64(5):778–791
25. Collins K PASTA for proteins github site. https://github.com/kodicollins/pasta-databases
26. Nute M (2019) Github site for PASTA+BAli-Phy. https://github.com/mgnute/pasta. Accessed 18 July 2019
27. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological sequence analysis. Cambridge University Press, Cambridge
28. Warnow T (2018) Computational phylogenetics: an introduction to designing methods for phylogeny estimation. Cambridge University Press, Cambridge
29. Eddy SR (2009) A new generation of homology search tools based on probabilistic inference. Genome Inform 23:205–211
30. Finn RD, Clements J, Eddy SR (2011) HMMER web server: interactive sequence similarity searching. Nucleic Acids Res 39: W29–W37
31. Novák Á, Miklós I, Lyngsoe R, Hein J (2008) StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. Bioinformatics 24:2403–2404
32. Huelsenbeck J, Ronquist R (2001) MrBayes: Bayesian inference of phylogeny. Bioinformatics 17:754–755
33. Drummond A, Rambaut A (2007) BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evol Biol 7:214
34. Bouckaert R, Heled J, Kühnert D, Vaughan T, Wu C-H, Xie D, Suchard MA, Rambaut A, Drummond AJ (2014) BEAST 2: a software platform for Bayesian evolutionary analysis. PLoS Comput Biol 10(4):e1003537
35. Lefort V, Desper R, Gascuel O (2015) FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program. Mol Biol Evol 32(10):2798–2800
36. Goloboff P, Farris J, Nixon K (2008) TNT, a free program for phylogenetic analysis. Cladistics 24:1–13
37. Swofford DL (1996) PAUP*: Phylogenetic analysis using parsimony (and other methods), Version 4.0. Sinauer Associates, Sunderland

38. Naser-Khdour S, Minh BQ, Zhang W, Stone E, Lanfear R (2019) The prevalence and impact of model violations in phylogenetics. BioRxiv. https://doi.org/10.1101/460121
39. Crotty SM, Minh BQ, Bean NG, Holland BR, Tuke J, Jermiin LS, Haeseler Av (2019) GHOST: recovering historical signal from heterotachously-evolved sequence alignments. bioRxiv, https://doi.org/10.1101/174789
40. Jermiin LS, Catullo RA, Holland BR (2018) A new phylogenetic protocol: dealing with model misspecification and confirmation bias in molecular phylogenetics. bioRxiv, https://doi.org/10.1101/400648
41. Nelesen S, Liu K, Wang L-S, Linder CR, Warnow T (2012) DACTAL: divide-and-conquer trees (almost) without alignments. Bioinformatics 28:i274–i282
42. Zhang Q, Rao S, Warnow T (2019) Constrained incremental tree building: new absolute fast converging phylogeny estimation methods with improved scalability and accuracy. Algorithms Mol Biol 14(1):2
43. Le T, Sy A, Molloy EK, Zhang QR, Rao S, Warnow T (2019) Using INC within divide-and-conquer phylogeny estimation. In: International conference on algorithms for computational biology. Springer, Berlin, pp 167–178
44. Molloy EK, Warnow T (2018) NJMerge: a generic technique for scaling phylogeny estimation methods and its application to species trees. In: RECOMB International conference on comparative genomics. Springer, Berlin, pp 260–276
45. Molloy EK, Warnow T (2019) TreeMerge: a new method for improving the scalability of species tree estimation methods. Bioinformatics 35(14):i417–i426
46. Sayyari E, Whitfield JB, Mirarab S (2017) Fragmentary gene sequences negatively impact gene tree and species tree Reconstruction. Mol. Biol. Evol. 34(12):3279–3291
47. Jarvis E, Mirarab S, Aberer AJ, Li B, Houde P, Li C, Ho S, Faircloth BC, Nabholz B, Howard JT, Suh A, Weber CC, daFonseca RR, Li J, Zhang F, Li H, Zhou L, Narula N, Liu L, Ganapathy G, Boussau B, Bayzid MS, Zavidovych V, Subramanian S, Gabaldón T, Capella-Gutiérrez S, Huerta-Cepas J, Rekepalli B, Munch K, Schierup M, Lindow B, Warren WC, Ray D, Green RE, Bruford MW, Zhan X, Dixon A, Li S, Li N, Huang Y, Derryberry EP, Bertelsen MF, Sheldon FH, Brumfield RT, Mello CV, Lovell PV, Wirthlin M, Schneider MPC, Prosdocimi F, Samaniego JA, Velazquez AMV, Alfaro-Núnez A, Campos PF, Petersen B, Sicheritz-Ponten T, Pas A, Bailey T, Scofield P, Bunce M, Lambert DM, Zhou Q, Perelman P, Driskell AC, Shapiro B, Xiong Z, Zeng Y, Liu S, Li Z, Liu B, Wu K, Xiao J, Yinqi X, Zheng Q, Zhang Y, Yang H, Wang J, Smeds L, Rheindt FE, Braun M, Fjeldsa J, Orlando L, Barker FK, Jonsson KA, Johnson W, Koepfli K-P, O'Brien S, Haussler D, Ryder OA, Rahbek C, Willerslev E, Graves GR, Glenn TC, McCormack J, Burt D, Ellegren H, Alstrom P, Edwards SV, Stamatakis A, Mindell DP, Cracraft J, Braun EL, Warnow T, Jun W, Gilbert MTP, Zhang G (2014) Whole-genome analyses resolve early branches in the tree of life of modern birds. Science 346 (6215):1320–1331
48. Do CB, Gross SS, Batzoglou S (2006) CONTRAlign: discriminative training for protein sequence alignment. In: Proceedings of the tenth annual international conference on computational molecular biology (RECOMB 2006). Springer, Berlin, pp 160–174
49. Do CB, Mahabhashyam MS, Brudno M, Batzoglou S (2005) ProbCons: probabilistic consistency-based multiple sequence alignment Genome Res 15(2):330–340
50. Do CB, Mahabhashyam MS, Brudno M, Batzoglou S (2006) ProbCons: probabilistic consistency-based multiple sequence alignment of amino acid sequences. Software available at http://probcons.stanford.edu/download.html
51. Liu K, Linder C, Warnow T (2012) RAxML and FastTree: comparing two methods for large-scale maximum likelihood phylogeny estimation. PLoS ONE 6(11):e27731
52. Abascal F, Zardoya R, Posada D (2005) ProtTest: selection of best-fit models of protein evolution. Bioinformatics 21(9):2104–2105
53. Posada D, Crandall K (1998) Modeltest: testing the model of DNA substitution. Bioinformatics 14(9):817–818
54. Hoff M, Orf S, Riehm B, Darriba D, Stamatakis A (2016) Does the choice of nucleotide substitution models matter topologically? BMC Bioinformatics 17:143
55. Tavaré S (1986) Some probabilistic and statistical problems in the analysis of DNA sequences. In: Lectures on mathematics in the life sciences, vol 17. American Mathematical Society, Providence, pp 57–86
56. Jukes TH, Cantor CR (1969) Evolution of protein molecules. In: Munro HN (ed) Mammalian protein metabolism. Academic, New York, pp 21–132
57. Nute M, Warnow T (2016) Scaling statistical multiple sequence alignment to large datasets. BMC Genomics 17:764(2016) Special issue for RECOMB-CG 2016. https://doi.org/10.1186/s12864-016-3101-8

58. Nute M, Saleh E, Warnow T (2018) Evaluating statistical multiple sequence alignment in comparison to other alignment methods on protein data sets. Syst Biol 68(3):396–411
59. Bateman A, Birney E, Cerruti L, Durbin R, Etwiller L, Eddy SR, Griffiths-Jones S, Howe KL, Marshall M, Sonnhammer EL (2002) The Pfam protein families database. Nucleic Acids Res. 30:276–280
60. Mai U, Mirarab S (2018) TreeShrink: fast and accurate detection of outlier long branches in collections of phylogenetic trees. BMC Genomics 19(S5):272

Chapter 8

Sequence Comparison Without Alignment: The *SpaM* Approaches

Burkhard Morgenstern

Abstract

Sequence alignment is at the heart of DNA and protein sequence analysis. For the data volumes that are nowadays produced by massively parallel sequencing technologies, however, pairwise and multiple alignment methods are often too slow. Therefore, fast alignment-free approaches to sequence comparison have become popular in recent years. Most of these approaches are based on *word frequencies*, for words of a fixed length, or on word-*matching* statistics. Other approaches are using the length of *maximal word matches*. While these methods are very fast, most of them rely on ad hoc measures of sequences similarity or dissimilarity that are hard to interpret. In this chapter, I describe a number of alignment-free methods that we developed in recent years. Our approaches are based on *spaced-word matches ("SpaM")*, i.e. on inexact word matches, that are allowed to contain mismatches at certain pre-defined positions. Unlike most previous alignment-free approaches, our approaches are able to accurately estimate phylogenetic distances between DNA or protein sequences using a stochastic model of molecular evolution.

Key words Genome comparison, Alignment free, Phylogeny, SpaM, Phylogenomics, Spaced words, FSWM

1 Introduction

Alignment-free sequence comparison has a long tradition in bioinformatics. The first approaches to compare sequences without alignments were proposed in the Nineteen-eighties by E. Blaisdell [1, 2]. The interest in alignment-free methods increased when more and more partially or completely sequenced genomes became available through novel sequencing technologies, leading to an urgent need for faster methods of sequence comparison. Most existing alignment-free methods represent sequences as *word-frequency vectors* for words of a fixed length *k*—so-called *k*-mers—and by comparing *k*-mer frequencies instead of comparing sequences position-by-position, based on alignments [3–7]. This approach has been further elaborated by taking background probabilities of word matches into account [8–11]; a review of these latter methods is given in [12]. Other approaches to alignment-free sequence comparison use the length of *maximal common sub-words* of the compared sequences, to define alternative measures of sequence similarity or dissimilarity [13–17].

The main advantage of these word-based methods is their high speed, compared to alignment-based methods. While—under most scoring schemes—finding an optimal alignment of two sequences takes time proportional to the *product* of their lengths [18–20], word-based or alignment-free methods are much more efficient, since word-frequency vectors can be calculated in time proportional to the length of the analyzed sequences. Similarly, the length of longest common sub-words can be efficiently found using data structures such as *generalized suffix trees* or *suffix arrays* [21]. A review of earlier alignment-free methods is given in [22]; more recent review papers are [23–27]. A first systematic benchmark study of alignment-free methods has been published in 2019, as a collaboration of several groups working in the field [28].

From the beginning, *phylogenetic tree reconstruction* has been a main application of alignment-free sequence comparison. Choi and Kim, for example, were able to calculate a phylogenetic tree of >4000 whole-proteome sequences [29], using the alignment-free tool *FFP* that has been developed by the same group [5]. The fastest tree-reconstruction methods are *distance-based* approaches: to calculate a tree representing the phylogeny of a set of taxa, these methods use pairwise distances as input, so for each pair of compared taxa, their distance or dissimilarity needs to be measured in some way. A matrix with these distance values can then be used as input for standard distance methods such as *Neighbor-Joining (NJ)* [30] or *BIONJ* [31].

If DNA sequences are compared, a common way of defining the distance between two evolutionarily related sequences is to use the (estimated) number of *substitutions per position* that have occurred since the two sequences have evolved from their last common ancestor. The simplest substitution model for nucleic-acid sequences is the *Jukes–Cantor* model where all nucleotide substitutions are assumed to occur with the same probability. Under this model, the number of *substitutions per position* can be estimated from the *number of mismatches per position* in an alignment of the compared sequences, using the well-known *Jukes–Cantor formula* [32]. More elaborate substitution models are available for DNA or protein sequences, that consider different substitution probabilities for different pairs of nucleotide or amino-acid residues.

A draw-back of most earlier alignment-free methods is that they are not based on probabilistic models of evolution. Instead, they use heuristic measures of sequence similarity or dissimilarity. If sequences are represented by word-frequency vectors, for example, standard distance measures on vector spaces can be applied to these frequency vectors, in order to calculate a "distance" between two compared sequences, such as the *Euclidean distance* or the *Manhattan distance*. Such distances, however, are hard to interpret from a phylogenetic point-of-view. Clearly, a pair of closely related sequences will have more words in common, compared to a pair of distantly related sequences—so the Euclidean distance between their word-frequency vectors will be smaller than for sequences that are further apart in the tree of life. But distance values calculated in this way do not represent real distances in terms of events that have happened since two sequences evolved from their last common ancestor. They only indicate if one pair of sequences is more or less similar to each other than another pair of sequences. Such heuristic distance measures can be used for clustering, but not for more accurate phylogenetic analyses.

Since the distances calculated by standard word-based methods have no direct phylogenetic interpretation, it would make no sense to "evaluate" the accuracy of these distance values directly. The developers of earlier alignment-free methods therefore took an *indirect* approach to evaluate their methods. They applied *clustering* algorithms or distance-based tree-reconstruction methods to the distances produced by the various alignment-free methods, and evaluated the quality of the resulting trees. Again, since the computed distances between the sequences have no direct meaning, the branch-lengths of these trees were usually ignored, and only the resulting *tree topologies* were evaluated, i.e. the order in which the taxa branched from each other in their history. The standard approach to evaluate tree topologies is to compare them to trusted reference *topologies* under the *Robinson–Foulds* metric [33]. Clearly, this is only a very rough way of evaluating the performance of sequence comparison methods.

Only in the last 10 years or so, alignment-free methods have been proposed that are able to estimate phylogenetic distances in the sense of an underlying probabilistic model of sequence evolution. The first such approach has been published in 2009 by Haubold et al. [34]. These authors developed *kr*, an alignment-free method that can accurately estimate phylogenetic distances between *DNA* sequences in the sense of the *Jukes–Cantor* model. That is, *kr* estimates the number of nucleotide substitutions per sequence position since the compared sequences have evolved from their last common ancestor. To this end, the program used the average length of common substrings between the compared sequences. Later, we proposed an approach to estimate phylogenetic distances based on the length distribution of *k*-mismatch common substrings [35].

In the last few years, other alignment-free methods have been proposed to estimate phylogenetic distances in a rigorous way [36–39]. Some of these methods are based on the so-called *micro-alignments*, short gap-free pairwise alignments with a simplistic structure, that can be rapidly calculated. So, strictly spoken, these methods are not quite *alignment-free*. They are referred to as "alignment-free" anyway, since they avoid the time-consuming process of calculating optimal alignments over the entire length of the input sequences. Other approaches to estimate distances in a stochastically rigorous way are based on the number of word matches [40]. More recently, extremely fast programs have become popular that can accurately estimate phylogenetic distances between DNA sequences from the number of word matches, using the *Jaccard Index* [41] and *min-hash* algorithms [42]. A widely used implementation of these ideas is *Mash* [43]; further improvements to this approach have been proposed and are implemented in the programs *Skmer* [44], *Dashings* [45], and *Mash Screen* [46].

2 Spaced Words

In 2013, we proposed to use the so-called *spaced words* for alignment-free *DNA* and protein sequence comparison [47–49]. A spaced word is a word that contains not only nucleotide or amino-acid symbols, but also *wildcard* characters at certain positions. A spaced word is based on a pre-defined binary pattern P representing *match positions* ("1") and *don't-care positions* ("0"). Given such a pattern P, we defined a spaced word w with respect to P as a word that has the same length as the pattern P and that has symbols representing nucleotide or amino-acid residues at the *match positions* of P and the *wildcard symbol* ("$*$") at the *don't-care positions*, *see* Fig. 1 for an example. Spaced words—or *spaced seeds*—have been previously introduced in database searching, to improve the sensitivity of the standard *seed-and-extend* search strategy [50]. Efficient algorithms have been proposed to optimize the underlying patterns [51], and for *spaced-seed hashing* [52].

In a first study, we simply replaced standard *word frequencies* by *spaced-word frequencies*, to calculate distances between DNA and protein sequences [47]. As in earlier word-based methods, we used the *Euclidean* distance or, alternatively, the *Jensen-Shannon* distance between spaced-word frequency vectors to define the

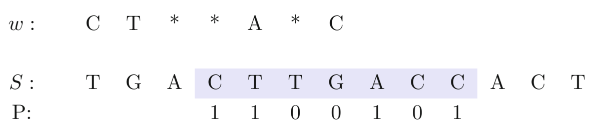


Fig. 1 Spaced word w with respect to a pattern $P = 1100101$ of length $\ell = 7$. w consists of nucleotide symbols at the *match positions* ("1") of P and of wildcard symbols, represented as "$*$" at the *don't-care positions* ("0"). w occurs at position 4 in the *DNA* sequence S

P :			1	1	0	0	1	0	1				
S_1 :	T	G	C	T	T	G	A	C	C	A	C	T	C
S_2 :	A	C	G	C	T	C	G	A	T	C	G	A	
P :				1	1	0	0	1	0	1			

Fig. 2 *Spaced-word match (SpaM)* between two DNA sequences S_1 and S_2 with respect to the same pattern P as in Fig. 1. The two segments have matching nucleotides at all *match positions* of P but may mismatch at the *don't-care* positions

distance between two DNA or protein sequences. This way, we could improve the quality the resulting phylogenetic trees, compared to standard word-frequency methods—in particular when we used *multiple* binary patterns and the corresponding spaced-word frequencies, instead of one single pattern [48]. The resulting software program is called *Spaced Words*, or *Spaced*, for short.

Our spaced-words approach was motivated by the *spaced seeds* [53] that have been introduced in database searching, to improve the sensitivity of *hit-and-extend* approaches such as *BLAST* [54]. The main advantage of spaced-word matches—or "spaced seeds"—compared to contiguous word matches is that neighboring spaced-word matches are statistically less dependent, so they are distributed more evenly over the sequences. In database searching, this increases the *sensitivity*, i.e. the probability of finding sequence similarities. For alignment-free sequence comparison, we have shown that results obtained with spaced words are statistically more stable than results based on contiguous words [40].

Note, however, that, like earlier alignment-free approaches, this first version of the program *Spaced* was still based on a heuristic measure of sequence dissimilarity; it did not estimate evolutionary distances in the sense of some probabilistic model. Later, we introduced a new distance measure in *Spaced* based on the number of spaced-word matches [40] that actually estimates phylogenetic distances between DNA sequences in the sense of the *Jukes–Cantor* model [32]. More precisely, a spaced-word match at positions (i, j) between two input sequences is the occurrence of the same spaced word w at position i in the first sequence and at position j in the second sequences, *see* Fig. 2. Our program calculates the number of pairs (i, j) for which there is a spaced-word match at (i, j). This is now the default distance measure used in the program *Spaced*. To find good patterns—or *sets* of patterns in the *multiple-pattern* approach—we developed a program called *rasbhari* [55].

3 *Filtered Spaced-Word Matches* and *Prot-SpaM*

In a subsequent project, we introduced a different approach to use spaced words for alignment-free sequence comparison. Instead of comparing spaced-word *frequencies*, we used *spaced-word matches*

(SpaM) as a special type of *micro-alignments*. For a binary pattern *P* as above, a *SpaM* between two sequences is simply the occurrence of the same spaced word in both sequences with respect to *P*, *see* Fig. 2 for an example. In other words, a *SpaM* is a local, gap-free alignment that has the same length as the pattern *P* and that has matching nucleotides or amino acids at the *match positions* and possible mismatches at the *don't-care positions* of *P*. The idea is to consider a large number of *SpaMs*, and to estimate phylogenetic distances between two sequences by looking at the residues that are aligned to each other at the *don't-care positions* of these *SpaMs*. Obviously, this is only possible if the considered *SpaMs* represent true homologies, so one has to filter out spurious background *SpaMs*. To do so, our program first considers all *possible SpaMs* between two input sequences and calculate a score for each *SpaM* based on the aligned residues at the *don't-care* positions. The program then discards all *SpaMs* with scores below some threshold. We could show that, with this sort of *"SpaM filter,"* one can reliably separate true homologies ("signal") from random *SpaM* ("noise").

An implementation of this approach for *DNA* sequences is called *Filtered Spaced-Word Matches (FSWM)*. To estimate distances between *DNA* sequences, *FSWM* calculates the proportion of mismatches at the *don't-care* positions of the selected *SpaMs*, as an estimate of the proportion of mismatches in the (unknown) full alignment of the two sequences. It then applies the usual *Jukes–Cantor* correction, to calculate the estimated *number of substitutions per position* since the two sequences have evolved from their last common ancestor. By default, the program uses a pattern *P* of length $\ell = 112$ with 12 *match positions* and 100 *don't-care* positions, but the user can adjust these parameters. The length of the pattern *P* seems to be a certain limitation, as it means that, by default, the program is restricted to using gap-free homologies of length $\geq \ell = 112$. A sufficient number of *don't-care* positions is necessary, though, to reliably distinguish *SpaMs* representing true homologies from random background *SpaMs*. To speed-up the program, it can be run with multiple threads; by default 10 threads are used. More recently, we evaluated different *sampling* strategies, to reduce the number of *SpaMs* that need to be evaluated during a program run [56].

An implementation of the same algorithm for protein sequences is called *Prot-SpaM* [57]. This program uses, by default, a set of 5 patterns with 6 *match positions* and 40 *don't-care positions* each, i.e. with a length of $\ell = 46$. For protein sequences, we are using the *BLOSUM 64* substitution matrix [58] to score *SpaMs*, and to filter out low-scoring random *SpaMs*. To estimate the evolutionary distance between two protein sequences, pairs of amino acids aligned to each other at the don't-care positions of the selected spaced-word matches are considered, and the *Kimura* model [59] is used to approximate the *PAM* distance [60] between sequences based on the number of mismatches per position. If desired, the user can use different values for the number of *match positions* in the underlying patterns *P*, and a different length for the patterns. The threshold for the *score* of the spaced-word matches is set to 0 by default, but can also be adapted and, similarly, the number of patterns can be modified.

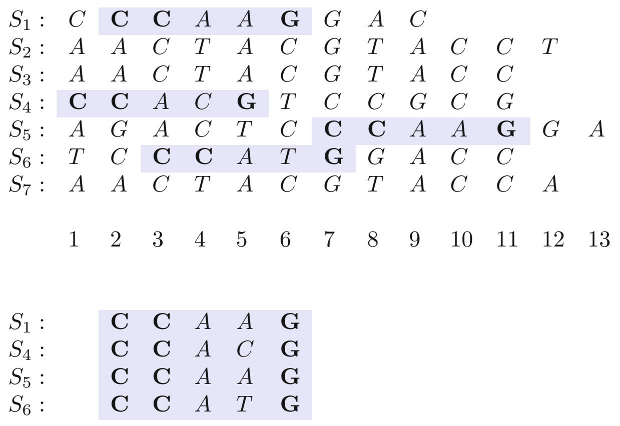
4 *Read-SpaM:* Estimating Phylogenetic Distances Based on Unassembled Sequencing Reads

Several authors have pointed out that alignment-free approaches can be applied, in principle, not only to full genome sequences, but also to unassembled reads. Some approaches have been designed for this particular purpose [36, 44]. The ability to estimate phylogenetic distances based on unassembled reads is not only useful in phylogeny studies, but also in biodiversity research [44] or in clinical studies [61, 62]. Here, species or strains of bacteria can often be identified by *genome skimming*, i.e. by low-coverage sequencing [44, 63–67].

We adapted *FSWM* to estimate phylogenetic distances between different taxa using unassembled reads; we called this approach *Read-SpaM* [68]. This software can estimate distances between an assembled genome from one taxon and a set of unassembled reads from another taxon or between sets of unassembled reads from two different taxa. Using simulated sequence data, we could show that *Read-SpaM* can accurately estimate distances between genomes up to 0.8 substitutions per position, for a sequence coverage as low as $2^{-9}X$, if an assembled genome is compared to a set of unassembled reads from a second genome. If sets of unassembled reads from two different taxa are compared to each other, distances estimates by *Read-SpaM* are still accurately for up to 0.7 substitutions per position, for a sequencing coverage down to $2^{-4}X$.

5 The Most Recent Approaches: *Multi-SpaM* and *Slope-SpaM*

For nucleic-acid sequences, we extended *Filtered Spaced Words Matches* from pairwise to multiple sequence comparison [69]. For a set of $N \geq 4$ input sequences, our software *Multi-SpaM* is based on spaced-word matches between four input sequences each. Such a multiple spaced-word match is, thus, a local gap-free *four-way alignment* with columns of identical nucleotides at the *match positions* of the underlying binary pattern *P*, while mismatches are, again, allowed at the *don't-care positions* of *P*. An example is given in Fig. 3, such local four-way alignments are also called *P-blocks.*



S_1: C C C A A G G A C
S_2: A A C T A C G T A C C T
S_3: A A C T A C G T A C C
S_4: C C A C G T C C G C G
S_5: A G A C T C C C A A G G A
S_6: T C C C A T G G A C C
S_7: A A C T A C G T A C C A
1 2 3 4 5 6 7 8 9 10 11 12 13
S_1: C C A A G
S_4: C C A C G
S_5: C C A A G
S_6: C C A T G

Fig. 3 *P*-block for a pattern $P = 11001$: a spaced word $W = CC{*}{*}G$ with respect to P occurs in sequences S_1, S_4, S_5, and S_6 at positions 2, 1, 7, and 3, respectively (top). Such a *P*-block defines a gap-free local four-way alignment with matching nucleotides at the *match positions* of the underlying pattern *P* and possible mismatches at the *don't-care positions* (bottom)

Multi-SpaM samples *P*-blocks from the set of input sequences; by default up to 10^6 *P*-blocks are sampled. To ensure that these *P*-blocks represent true homologies, only those *P*-blocks are considered that have a score above a certain threshold. For each of the sampled *P*-blocks, the program then uses *RAxML* [70] to calculate an optimal unrooted quartet tree topology. Finally, the program *Quartet MaxCut* [71] is used to calculate a super tree topology from these quartet topologies. By default, *Multi-SpaM* uses a pattern *P* with a length of 110 nucleotides and with 10 *match positions*, i.e. with 100 *don't-care* positions. These parameters can be adjusted by the user. Also, the number of 10^6 *P*-blocks that are sampled is a parameter that can be adjusted by the user.

As another approach to alignment-free phylogeny reconstruction, we developed a program called *Slope-SpaM* [72]. This program considers the number N_k of *k*-mer matches—or *spaced-word* matches for a pattern P_k with *k match positions*, respectively—between two nucleic-acid sequences, for different values of *k*. As we have shown theoretically, one can define a function *F* where $F(k)$ depends on N_k, such that *F* is affine-linear in a certain range of *k*. The *Jukes–Cantor* distances between the two sequences—i.e. the average number of substitutions per sequence position since the sequences diverged from their most recent common ancestor—can then be estimated from the slope of *F* within this range. In addition, we showed in [72], how one can calculate two values $k_{\min}$ and $k_{\max}$ within the relevant affine-linear range. The slope of *F* in this range—and therefore the distance between the two sequences—can, thus, be estimated from the values $N_{k_{\min}}$ and $N_{k_{\max}}$ alone, the program is therefore extremely fast. This way, evolutionary distances can be calculated accurately for up to around 0.5 substitutions per sequence position.

Several other methods have been proposed in recent years that estimate evolutionary distances from the number of *k*-mer or spaced-word matches [40, 44, 73]. A limitation of these methods is that they assume that the compared sequences are homologous over their entire length; they under-estimate distances if sequences share only local homologies. In contrast, *Slope-SpaM* can estimate phylogenetic distances even if sequences share only *local* regions of homology.

By default, *Slope-SpaM* uses exact word matches and calculates the values $N_{k_{\min}}$ and $N_{k_{\max}}$ based on the length of the input sequences. The user can also specify a binary pattern P with a sufficiently large number k_M of *match positions* ("1"). Patterns P_k with smaller numbers of *match positions* are then generated by shortening P_{k_M}. Instead of calculating $N_{k_{\min}}$ and $N_{k_{\max}}$ automatically, *Slope-SpaM* can also take a set of values of k as input. It will then calculate the function $F(k)$ for each specified value of k and find the slope of the affine-linear region by linear regression.

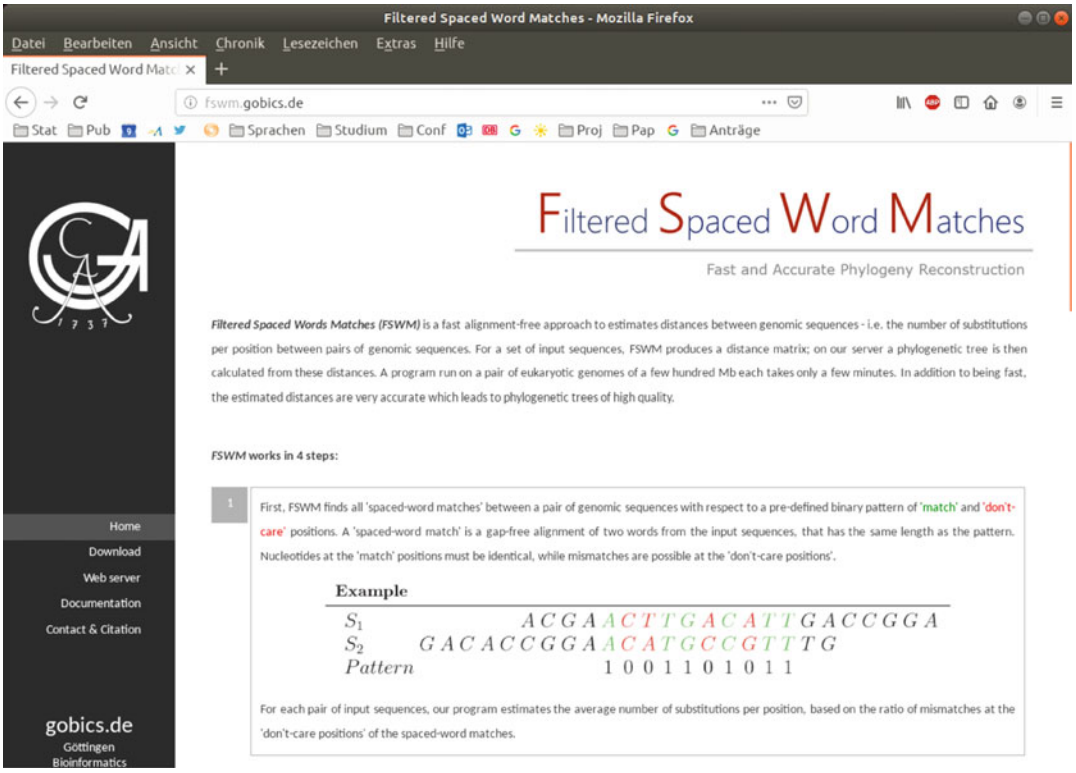
6 Back to Multiple Sequence Alignment

There is no strict separation between sequence alignment on one side and word-based, alignment-free methods on the other side. As mentioned above, a whole class of the so-called alignment-free methods are based on "micro-alignments," local pairwise alignments of a simple structure, that can be rapidly calculated. In *Multi-SpaM*, we extended this approach to local *multiple* alignments.

Ironically, one of the first major applications of fast alignment-free methods was *multiple sequence alignment (MSA)*. The programs *MUSCLE* [74] or *Clustal Omega* [75], for example, use word-frequency vectors to rapidly calculate *guide trees* for the "progressive" approach to MSA [74]. Similarly, fast alignment-free methods are used to find *anchor points* [76, 77] to make alignments of large genomic sequences possible [78–84]. In a recent study [85], we used *FSWM* to generate anchor points for multiple genome alignment. We could show that, if distantly related genomes are compared, spaced-word matches are more sensitive and lead to better output alignments than anchor points that are based on exact word matches.

7 Software Availability

We made *Filtered Spaced-Word Matches (FSWM)* available through a web interface at *Göttingen Bioinformatics Compute Server (GOBICS)*, http://fswm.gobics.de/ *see* Fig. 4. There are certain



Filtered Spaced Word Matches - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Filtered Spaced Word Matc
fswm.gobics.de
Stat Pub Sprachen Studium Conf Proj Pap Anträge
Home
Download
Web server
Documentation
Contact & Citation
gobics.de
Göttingen Bioinformatics

Filtered Spaced Word Matches

Fast and Accurate Phylogeny Reconstruction

Filtered Spaced Words Matches (FSWM) is a fast alignment-free approach to estimates distances between genomic sequences - i.e. the number of substitutions per position between pairs of genomic sequences. For a set of input sequences, FSWM produces a distance matrix; on our server a phylogenetic tree is then calculated from these distances. A program run on a pair of eukaryotic genomes of a few hundred Mb each takes only a few minutes. In addition to being fast, the estimated distances are very accurate which leads to phylogenetic trees of high quality.

FSWM works in 4 steps:

1 First, FSWM finds all 'spaced-word matches' between a pair of genomic sequences with respect to a pre-defined binary pattern of 'match' and 'don't-care' positions. A 'spaced-word match' is a gap-free alignment of two words from the input sequences, that has the same length as the pattern. Nucleotides at the 'match' positions must be identical, while mismatches are possible at the 'don't-care positions'.

Example

```
S1              A C G A A C T T G A C A T T G A C C G G A
S2        G A C A C C G G A A C A T G C C G T T T G
Pattern                   1 0 0 1 1 0 1 0 1 1
```

For each pair of input sequences, our program estimates the average number of substitutions per position, based on the ratio of mismatches at the 'don't-care positions' of the spaced-word matches.

Fig. 4 Homepage of *Filtered Spaced-Word Matches (FSWM)* at *Göttingen Bioinformatics Compute Server (GOBICS)*

limitations at this web server for the size of the input data: (*a*) the upper limit for the total size of the input sequences is 512 mb, (*b*) the number of input sequences must be between 2 and 100, and (*c*) the minimum length of each input sequence is 1000 bp. At our web server, the underlying pattern P has by default 12 *match positions* and 100 *don't care positions*. The number of *match positions* can be adapted by the user. To calculate a *score* for each spaced-word match, a nucleotide substitution matrix published by Chiaromonte et al. [86] is used. By default, the cut-off value to distinguish "homologous" from background spaced-word matches is set to 0. This value, too, can be adjusted by the user.

In addition, the above described software tools *FSWM*, *Prot-SpaM*, *Multi-SpaM*, *Read-SpaM*, and *Slope-SpaM* are freely available as source code through *github* or through our home page, details are given in Table 1.

Table 1
Our software is available as open source code from the following URLs

Spaced Words	[40]	http://spaced.gobics.de/
FSWM	[38]	http://fswm.gobics.de/
Prot-SpaM	[57]	https://github.com/jschellh/ProtSpaM
Multi-SpaM	[69]	https://github.com/tdencker/multi-SpaM
Read-SpaM	[68]	https://github.com/burkhard-morgenstern/Read-SpaM
Slope-SpaM	[72]	https://github.com/burkhard-morgenstern/Slope-SpaM

References

1. Blaisdell BE (1986) A measure of the similarity of sets of sequences not requiring sequence alignment. Proc Natl Acad Sci USA 83:5155–5159
2. Blaisdell BE (1989) Average values of a dissimilarity measure not requiring sequence alignment are twice the averages of conventional mismatch counts requiring sequence alignment for a computer-generated model system. J Mol Evol 29:538–547
3. Teeling H, Meyerdierks A, Bauer M, Amann R, Glöckner FO (2004) Application of tetranucleotide frequencies for the assignment of genomic fragments. Environ Microbiol 6:938–947
4. Höhl M, Rigoutsos I, Ragan MA (2006) Pattern-based phylogenetic distance estimation and tree reconstruction Evol Bioinform Online 2:359–375
5. Sims GE, Jun S-R, Wu GA, Kim S-H (2009) Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. Proc Natl Acad Sci 106:2677–2682
6. Chor B, Horn D, Levy Y, Goldman N, Massingham T (2009) Genomic DNA *k*-mer spectra: models and modalities. Genome Biol 10: R108
7. Vinga S, Carvalho AM, Francisco AP, Russo LMS, Almeida JS (2012) Pattern matching through Chaos Game Representation: bridging numerical and discrete data structures for biological sequence analysis. Algorithm Mol Biol 7:10
8. Reinert G, Chew D, Sun F, Waterman MS (2009) Alignment-free sequence comparison (I): statistics and power. J Comput Biol 16:1615–1634
9. Wan L, Reinert G, Sun F, Waterman MS (2010) Alignment-free sequence comparison (II): theoretical power of comparison statistics. J Comput Biol 17:1467–1490
10. Song K, Ren J, Zhai Z, Liu X, Deng M, Sun F (2013) Alignment-free sequence comparison based on next-generation sequencing reads. J Comput Biol 20:64–79
11. Ahlgren NA, Ren J, Lu YY, Fuhrman JA, Sun F (2017) Alignment-free d_2^* oligonucleotide frequency dissimilarity measure improves prediction of hosts from metagenomically-derived viral sequences. Nucleic Acids Res 45:39–53
12. Ren J, Bai X, Lu YY, Tang K, Wang Y, Reinert G, Sun F (2018) Alignment-free sequence analysis and applications. Annu Rev Biomed Data Sci 1:93–114
13. Ulitsky I, Burstein D, Tuller T, Chor B (2006) The average common substring approach to phylogenomic reconstruction. J Comput Biol 13:336–350
14. Comin M, Verzotto D (2012) Alignment-free phylogeny of whole genomes using underlying subwords. Algorithm Mol Biol 7:34
15. Leimeister C-A, Morgenstern B (2014) *kmacs*: the *k*-mismatch average common substring approach to alignment-free sequence comparison. Bioinformatics 30:2000–2008
16. Pizzi C (2016) MissMax: alignment-free sequence comparison with mismatches through filtering and heuristics. Algorithm Mol Biol 11:6
17. Thankachan SV, Chockalingam SP, Liu Y, Aluru AKS (2017) A greedy alignment-free distance estimator for phylogenetic inference BMC Bioinformatics 18:238
18. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48:443–453

19. Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological sequence analysis. Cambridge University Press, Cambridge
20. Morgenstern B (2000) A space-efficient algorithm for aligning large genomic sequences Bioinformatics 16:948–949
21. Gusfield D (1997) Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, Cambridge
22. Vinga S, Almeida J (2003) Alignment-free sequence comparison - a review Bioinformatics 19:513–523
23. Haubold B (2014) Alignment-free phylogenetics and population genetics Brief Bioinform 15:407–418
24. Song K, Ren J, Reinert G, Deng M, Waterman MS, Sun F (2014) New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. Brief Bioinform 15:343–353
25. Zielezinski A, Vinga S, Almeida J, Karlowski WM (2017) Alignment-free sequence comparison: benefits, applications, and tools. Genome Biol 18:186
26. Bernard G, Chan CX, Chan Y-B, Chua X-Y, Cong Y, Hogan JM, Maetschke SR, Ragan MA (2019) Alignment-free inference of hierarchical and reticulate phylogenomic relationships. Brief Bioinform 22:426–435
27. Kucherov G (2019) Evolution of biosequence search algorithms: a brief survey. Bioinformatics 35:3547–3552
28. Zielezinski A, Girgis HZ, Bernard G, Leimeister C-A, Tang K, Dencker T, Lau AK, Röhling S, Choi J, Waterman MS, Comin, M, Kim S-H, Vinga S, Almeida JS, Chan CX, James B, Sun F, Morgenstern B, Karlowski WM (2019) Benchmarking of alignment-free sequence comparison methods. Genome Biol 20:144
29. Choi J, Kim S-H (2020) Genome tree of life: deep burst of organism diversity. Proc Natl Acad Sci 117:3678–3686
30. Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4:406–425
31. Gascuel O (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Mol Biol Evol 14:685–695
32. Jukes TH, Cantor CR (1969) Evolution of protein molecules. Academy, New York
33. Robinson DF, Foulds L (1981) Comparison of phylogenetic trees. Math Biosci 53:131–147
34. Haubold B, Pfaffelhuber P, Domazet-Loso M, Wiehe T (2009) Estimating mutation distances from unaligned genomes. J Comput Biol 16:1487–1500
35. Morgenstern B, Schöbel S, Leimeister C-A (2017) Phylogeny reconstruction based on the length distribution of *k*-mismatch common substrings. Algorithm Mol Biol 12:27
36. Yi H, Jin L (2013) Co-phylog: an assembly-free phylogenomic approach for closely related organisms. Nucleic Acids Res 41:e75
37. Haubold B, Klötzl F, Pfaffelhuber P (2015) andi: fast and accurate estimation of evolutionary distances between closely related genomes. Bioinformatics 31:1169–1175
38. Leimeister C-A, Sohrabi-Jahromi S, Morgenstern B (2017) Fast and accurate phylogeny reconstruction using filtered spaced-word matches. Bioinformatics 33:971–979
39. Klötzl F, Haubold B (in press) Phylonium: fast estimation of evolutionary distances from large samples of similar genomes. Bioinformatics. https://doi.org/10.1093/bioinformatics/btz903
40. Morgenstern B, Zhu B, Horwege S, Leimeister C-A (2015) Estimating evolutionary distances between genomic sequences from spaced-word matches. Algorithm Mol Biol 10:5.
41. Jaccard P (1901) Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Société Vaudoise des Sciences Naturelles. 37:547–579
42. Broder A (1997) On the resemblance and containment of documents. In Proceedings of the compression and complexity of sequences 1997 SEQUENCES '97. IEEE Computer Society, Washington, DC, p 21
43. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, Phillippy AM (2016) Mash: fast genome and metagenome distance estimation using MinHash Genome Biol 17:132
44. Sarmashghi S, Bohmann K, Gilbert MTP, Bafna V, Mirarab S (2019) Skmer: assembly-free and alignment-free sample identification using genome skims. Genome Biol **20**:34
45. Baker DN, Langmead B (2019) Dashing: fast and accurate genomic distances with HyperLogLog. Genome Biol 20:265
46. Ondov BD, Starrett GJ, Sappington A, Kostic A, Koren S, Buck CB, Phillippy AM (2019) Mash Screen: high-throughput sequence containment estimation for genome discovery. Genome Biol 20:232
47. Boden M, Schöneich M, Horwege S, Lindner S, Leimeister C-A, Morgenstern B (2013) Alignment-free sequence comparison

with spaced *k*-mers. vol 34. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, Dagstuhl, pp 24–34

48. Leimeister C-A, Boden M, Horwege S, Lindner S, Morgenstern B (2014) Fast alignment-free sequence comparison using spaced-word frequencies. Bioinformatics 30:1991–1999
49. Horwege S, Lindner S, Boden M, Hatje K, Kollmar M, Leimeister C-A, Morgenstern B (2014) *Spaced words* and *kmacs*: fast alignment-free sequence comparison based on inexact word matches. Nucleic Acids Res 42: W7–W11
50. Li M, Ma B, Kisman D, Tromp J (2004) PatternHunter II: highly sensitive and fast homology search. J Bioinform Computat Biol 02:417–439
51. Ilie L, Ilie S, Bigvand AM (2011) SpEED: fast computation of sensitive spaced seeds. Bioinformatics 27:2433–2434
52. Petrucci E, Noé L, Pizzi C, Comin M (in press) Iterative spaced seed hashing: closing the gap between spaced seed hashing and *k*-mer hashing. J Comput Biol. https://doi.org/10.1089/cmb.2019.0298
53. Li M, Ma B, Kisman D, Tromp J (2003) PatternHunter II: highly sensitive and fast homology search. Genome Inform 14:164–175
54. Altschul SF (1989) Gap costs for multiple sequence alignment. J Theor Biol 138:297–309
55. Hahn L, Leimeister C-A, Ounit R, Lonardi S, Morgenstern B (2016) *rasbhari*: optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison. PLOS Comput Biol 12(10):e1005107
56. Elfmann C (2019) Implementation of sampling strategies for filtered spaced-word matches. Bachelor's thesis. University of Göttingen, Göttingen (August, 2019). Supervisor: B. Morgenstern
57. Leimeister C-A, Schellhorn J, Dörrer S, Gerth M, Bleidorn C, Morgenstern B (2019) Prot-SpaM: fast alignment-free phylogeny reconstruction based on whole-proteome sequences. GigaScience 8:giy148
58. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci USA 89:10915–10919
59. Kimura M (1983) The neutral theory of molecular evolution. Cambridge University Press, Cambridge
60. Dayhoff MO, Schwartz RM, Orcutt BC (1978) A model of evolutionary change in proteins. Atlas Protein Seq Struct 6:345–362
61. Deurenberg RH, Bathoorn E, Chlebowicz MA, Couto N, Ferdous M, García-Cobos S, Kooistra-Smid AM, Raangs EC, Rosema S, Veloo AC, Zhou K, Friedrich AW, Rossen JW (2017) Application of next generation sequencing in clinical microbiology and infection prevention. J Biotechnol 243:16–24
62. Břinda K, Callendrello A, Cowley L, Charalampous T, Lee RS, MacFadden DR, Kucherov G, O'Grady J, Baym M, Hanage WP (2018) Lineage calling can identify antibiotic resistant clones within minutes. bioRxiv:10.1101/403204
63. Weitemier K, Straub SCK, Cronn RC, Fishbein M, Schmickl R, McDonnell A, Liston A (2014) Hyb-seq: combining target enrichment and genome skimming for plant phylogenomics. Appl Plant Sci 2:1400042
64. Dodsworth S (2015) Genome skimming for next-generation biodiversity analysis. Trends Plant Sci 20:525–527
65. Richter S, Schwarz F, Hering L, Böggemann M, Bleidorn C (2015) The utility of genome skimming for phylogenomic analyses as demonstrated for glycerid relationships (Annelida, Glyceridae). Genome Biol Evol 7:3443–3462
66. Denver DR, Brown AMV, Howe DK, Peetz AB, Zasada IA (2016) Genome skimming: a rapid approach to gaining diverse biological insights into multicellular pathogens. PLOS Pathog 12(8):e1005713
67. Linard B, Arribas P, Andújar C, Crampton-Platt A, Vogler AP (2016) Lessons from genome skimming of arthropod-preserving ethanol. Mol Ecol Resour 16:1365–1377
68. Lau AK, Dörrer S, Leimeister C-A, Bleidorn C, Morgenstern B (2019) *Read-SpaM*: assembly-free and alignment-free comparison of bacterial genomes with low sequencing coverage. BMC Bioinform 20:638
69. Dencker T, Leimeister C-A, Gerth M, Bleidorn C, Snir S, Morgenstern B (2020) *Multi-SpaM*: a Maximum-Likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. NAR Genomics Bioinform 2:lqz013
70. Stamatakis A (2014) RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30:1312–1313
71. Snir S, Rao S (2012) Quartet MaxCut: a fast algorithm for amalgamating quartet trees. Mol Phylogenet Evol 62:1–8
72. Röhling S, Linne A, Schellhorn J, Hosseini M, Dencker T, Morgenstern B (2020) The number of *k*-mer matches between two DNA

sequences as a function of *k* and applications to estimate phylogenetic distances. PLoS One 15: e0228070

73. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, Phillippy AM (2016) Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol 17:132
74. Edgar RC (2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinform 5:113
75. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins DG (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7:539
76. Morgenstern B, Werner N, Prohaska SJ, Schneider RSI, Subramanian AR, Stadler PF, Weyer-Menkhoff J (2005) Multiple sequence alignment with user-defined constraints at GOBICS. Bioinformatics 21:1271–1273
77. Huang W, Umbach DM, Li L (2006) Accurate anchoring alignment of divergent sequences. Bioinformatics 22:29–34
78. Höhl M, Kurtz S, Ohlebusch E (2002) Efficient multiple genome alignment Bioinformatics 18:312S–320S
79. Morgenstern B, Rinner O, Abdeddaïm S, Haase D, Mayer K, Dress A, Mewes H-W (2002) Exon discovery by genomic sequence alignment. Bioinformatics 18:777–787
80. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL (2004) Versatile and open software for comparing large genomes. Genome Biol 5:R12+
81. Darling ACE, Mau B, Blattner FR, Perna NT (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. Genome Res 14:1394–1403
82. Darling AE, Mau B, Perna NT (2010) progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. PLoS One 5:e11147+
83. Angiuoli SV, Salzberg SL (2011) Mugsy: fast multiple alignment of closely related whole genomes. Bioinformatics 27:334–342
84. Paten B, Earl D, Nguyen N, Diekhans M, Zerbino D, Haussler D (2011) Cactus: algorithms for genome multiple sequence alignment Genome Res 21:1512–1528
85. Leimeister C-A, Dencker T, Morgenstern B (2019) Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points. Bioinformatics 35:211–218
86. Chiaromonte F, Yap VB, Miller W (2002) Scoring pairwise genomic sequence alignments. In Altman RB, Keith Dunker A, Hunter L, Klein TE (eds) Pacific symposium on biocomputing, Lihue, HI, pp. 115–126

Chapter 9

lamassemble: Multiple Alignment and Consensus Sequence of Long Reads

Martin C. Frith, Satomi Mitsuhashi, and Kazutaka Katoh

Abstract

Long DNA and RNA reads from nanopore and PacBio technologies have many applications, but the raw reads have a substantial error rate. More accurate sequences can be obtained by merging multiple reads from overlapping parts of the same sequence. lamassemble aligns up to ~1000 reads to each other, and makes a consensus sequence, which is often much more accurate than the raw reads. It is useful for studying a region of interest such as an expanded tandem repeat or other disease-causing mutation.

Key words MAFFT, LAST, last-train, Nanopore, PacBio, Paralogy, Dotplot, Genomic variations, Repeat expansion disease, Strand-asymmetric error pattern

1 Introduction

Long read single molecule sequencing technologies (nanopore and PacBio) are increasingly becoming used in medical research areas in the past few years. Major applications in human research are: sequencing genomes of pathogens in infectious diseases, detecting pathogenic genomic variants in cancer genomes or rare human Mendelian diseases [1–3], or resolving structural variations in human genomes [4]. For example, several new diseases caused by tandem repeat expansion were discovered in 2019. Most of these studies used long read sequencers to completely characterize the structure of the expanded repeat [5–8], which was difficult by conventional methods. One pathogenic expansion was identified by long read whole genome sequencing alone [9]. Long read sequencing has clear advantages in assessing tandem and interspersed repeats, because long enough reads can extend beyond the repeat into unique sequence, and has advantages in detecting changes in repeat regions, which are mutation-prone and may cause sequence rearrangements. Structural variations are usually described as deletion, duplication or insertion. However, there are more types of structural variations, e.g. chromothripsis. Annotation

	a	c	g	t
a	0.28	0.0011	0.013	0.0011
c	0.0015	0.20	0.00045	0.0027
g	0.027	0.00051	0.18	0.00086
t	0.0013	0.0033	0.00055	0.29

Base deletion probability: existence=0.037, extension=0.44
Base insertion probability: existence=0.034, extension=0.40

Fig. 1 Example of probabilities (i.e. rates) of substitution, deletion, and insertion between human DNA reads and a reference human genome. This is from human DNA sequenced with PromethION R9.4, base-called with Guppy 1.4.0, dataset ERR2631604 [11]. Read bases correspond to matrix columns and reference genome bases correspond to matrix rows. The 16 substitution probabilities sum to 1

of further complex rearrangements is still challenging but long read sequencing may help decipher them [10].

The raw DNA (or RNA) reads from these technologies have substantial error rates. The error rates vary with the precise version of sequencing technology and base-calling algorithm. Typically, different kinds of errors have different rates, e.g. nanopore reads often have a high rate of a↔g errors (Fig. 1). Note that these error rates are not strand-symmetric, e.g. the c↔t rates are much lower than the a↔g rates in Fig. 1. This makes it both tricky and useful to combine DNA reads from both strands of a DNA molecule: the high rate of a↔g errors can be compensated by the low rate of c↔t errors on the opposite strand. Finally, long reads tend to have relatively high rates of insertion and deletion errors, which are trickier to deal with than substitution errors.

In some scenarios, it is useful to obtain an accurate sequence, more accurate than the raw reads. For example, the precise sequence of an expanded tandem repeat correlates with the symptoms of neuronal intranuclear inclusion disease [9]: this could not be seen from the raw reads, but only from a more-accurate consensus sequence (Fig. 2, *see* **Note 1**).

lamassemble merges overlapping reads into a consensus sequence. Its method has two key ingredients. Firstly, it uses the mature multiple alignment tool MAFFT to align the reads to each other. Secondly, it uses the rate/probability of each error type (base deletions, base insertions, and each kind of substitution) to find the most probable alignments, and also to infer the most probable consensus sequence.

A similar software tool is pbdagcon (https://github.com/PacificBiosciences/pbdagcon). The main difference is that pbdagcon corrects errors in one read by comparing it to other reads, so it outputs a sequence that only covers one input read. In contrast, lamassemble can merge multiple overlapping reads into a consensus that covers all the input reads.

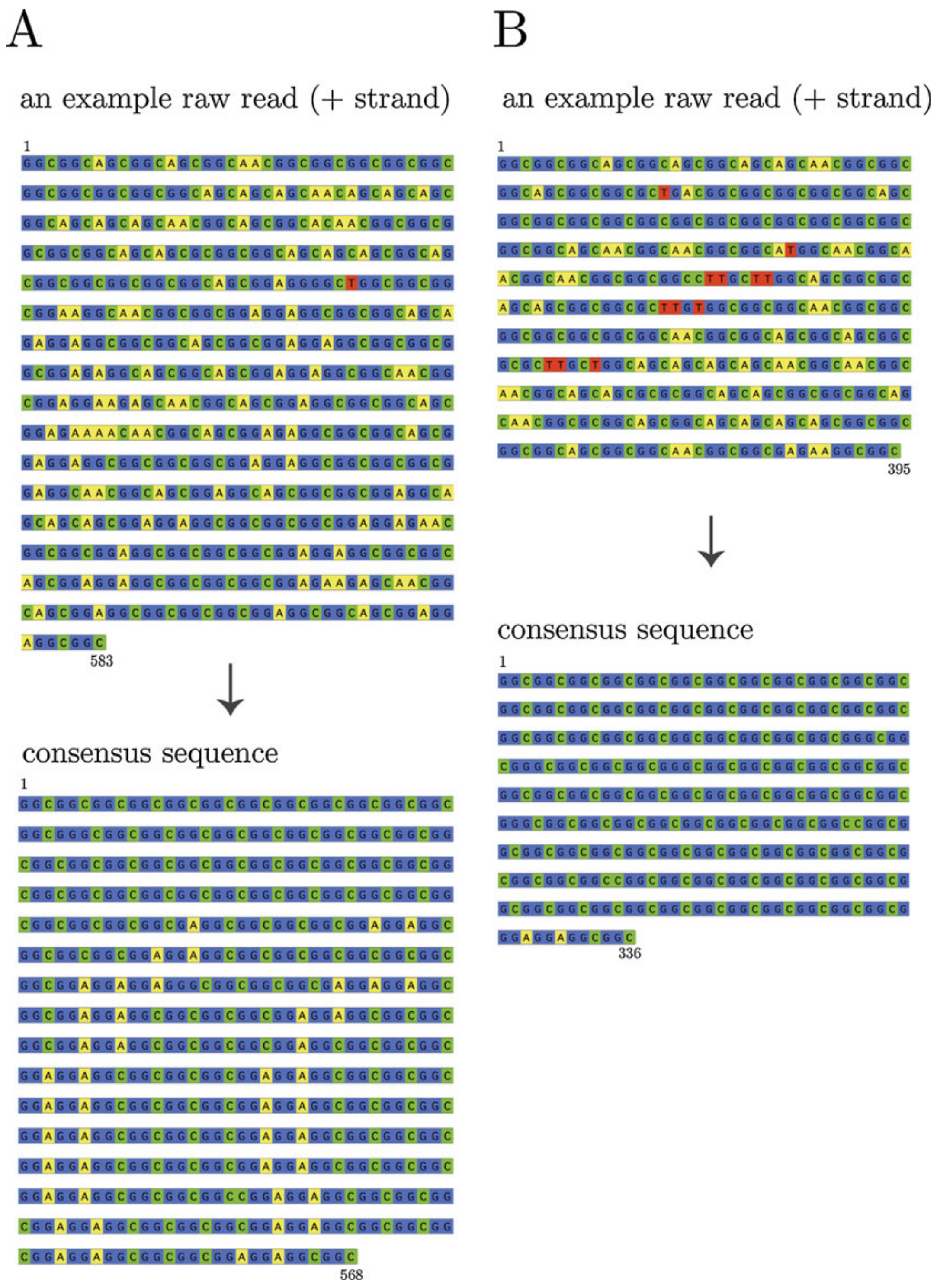

A
an example raw read (+ strand)
1
583
consensus sequence
1
568
B
an example raw read (+ strand)
1
395
consensus sequence
1
336

Fig. 2 Example of ggc repeat expansions in neuronal intranuclear inclusion disease [9]. (**a**) Example of ggc and gga repeat expansion. The consensus sequence was obtained from 61 nanopore raw reads that cover the whole expanded repeat. Raw reads have many errors. (**b**) Example of pure ggc repeat expansion. The consensus sequence was obtained from 448 nanopore reads that cover the whole expanded repeat. These consensus sequences are probably not 100% correct, e.g. their tandem repeats are disrupted by occasional base insertions or deletions which we suspect to be errors

2 Access/Installation

lamassemble can be used either via a web-server or by installing it on your own computer. The web-server is located at: https://mafft.cbrc.jp/alignment/server/index-rawreads.html.

Using it on your own computer requires using the "command line." It has been tested on Linux and Mac. If the Conda package manager is set up on your computer, lamassemble can be installed from the Bioconda channel, with this command:

```
conda install -c bioconda lamassemble
```

This automatically installs other software that lamassemble depends on (LAST and MAFFT). Alternatively, lamassemble can be obtained directly from its website: https://gitlab.com/mcfrith/lamassemble. In that case, as explained there, the dependencies must be installed manually.

3 Usage

3.1 Required Inputs for lamassemble

lamassemble requires two input files: a last-train file [12] (*see* **Notes 2** and **3**), and the sequences that we wish to merge (*see* **Notes 4**–7). The sequences may be in either fastq or fasta format: it makes no difference to the result, because lamassemble ignores the extra information present in fastq.

The last-train file indicates the error pattern of the reads, i.e. the rates of base deletion, base insertion, and each kind of substitution (e.g. a→g). last-train files for a few types of data are supplied with lamassemble: these are likely good enough for many datasets, especially for a quick first try.

Alternatively, a last-train file can be obtained by comparing the reads to a reference genome of the same (or very closely related) species, as described at: https://github.com/mcfrith/last-rna/blob/master/last-long-reads.md.

In our typical workflow, we first align a large set of reads to a reference genome, using last-train. We then pick out reads covering a region of interest, perhaps using tandem-genotypes for tandem repeat changes [13] or dnarrange for rearrangements [10]. Thus, by the time we wish to use lamassemble, we already have a last-train file for our data.

3.2 Command Line Usage

Typical command line usage is like this:

```
lamassemble last-train.mat sequences.fastq > consensus.fasta
```

This aligns the sequences in the file sequences.fastq to each other, and prints a consensus sequence, which we have sent to an output file named consensus.fasta.

3.3 Going Faster by Multi-Threading

The `-P` option makes it faster by running several threads of calculation concurrently. For example, `-P8` uses 8 threads:

```
lamassemble -P8 last-train.mat sequences.fastq > consensus.
fasta
```

The best number of threads depends on how many your computer can efficiently run concurrently. As a special case, `-P0` uses as many threads as your computer claims it can run concurrently.

3.4 Multiple Sequence Alignment

The `-a` option makes lamassemble output an alignment of the reads to each other:

```
lamassemble -a last-train.mat sequences.fastq > alignment.
fasta
```

4 How It Works

The detailed steps within lamassemble are described elsewhere [10]. It is not necessary for a user to understand this, but a rough understanding is useful for troubleshooting.

In a nutshell, lamassemble first runs the pairwise aligner LAST to find alignments between pairs of reads, then runs the multiple aligner MAFFT to align all the reads to each other, guided and constrained by the LAST alignments. Finally, it makes a consensus sequence from the multiple alignment.

The LAST step does not try to find alignments between every read and every other read: such a thorough approach would be feasible for a small number of reads, but the run time and memory use would explode quadratically as the number of reads increases. Instead, LAST aligns each position in each read to a few most-similar reads. The aim is to align every read to every other read either directly *or indirectly*.

lamassemble sorts the LAST alignments in descending order of score (indicating strength of similarity), and uses this order to define a guide tree for merging the sequences by MAFFT.

The LAST step also calculates an error probability for each pair of aligned bases in its alignments. There is more than one possible way to align a pair of sequences, and sometimes alternative alignments are nearly equally plausible (Fig. 3). Because LAST's alignments are based on probabilities of substitutions, deletions, and insertions, it is possible to calculate the probability that each pair of aligned bases is wrong [14]. lamassemble assumes that positions with error probability ≤ 0.002 are reliable, and passes them as "anchors" to MAFFT. (This 0.002 threshold can be adjusted by the user, but this seems to be rarely, if ever, necessary.)

```
gtatatgaattccaattcttaacccccctta
```

Fig. 3 Example of ambiguous alignment. Two different alignments of the same sequences are shown. `tagt` in the bottom sequence can be aligned to two different parts of the top sequence

These error probabilities have an important limitation: they indicate homology, including paralogy. For example, if a DNA molecule contains a recent tandem duplication, a DNA read from one copy of the duplication may align with low error probabilities to a DNA read from the other copy. These two copies are true homologs, because they are descended from a common ancestor. But they are paralogs (duplicates within a genome), and lamassemble should not merge them in order to accurately reconstruct the DNA molecule. It can be arbitrarily difficult to avoid conflating paralogs that are both large and recent (thus highly similar).

Pairwise alignment information calculated by LAST is passed to MAFFT. MAFFT aligns input sequences progressively, i.e., repeating group-to-group alignment calculation from the terminal nodes to the root on the guide tree. In a group-to-group alignment step, each of two groups of sequences is split into short subsequences using anchors found by LAST, where different pairs across the two groups are considered. There can be positionally inconsistent anchors for a pair of groups. To avoid this inconsistency, anchors from higher-scoring LAST alignments are selected with higher priority, while those with lower scores are excluded if inconsistent with any anchors with higher scores. Optimum combination of anchors is not searched for. Using this set of anchors, sequences in a pair of groups are consistently split into short subsequences. Each pair of groups of subsequences is aligned assuming the error rate estimated by last-train. Then, all aligned subsequences are concatenated to generate an aligned pair of groups of sequences. This group-to-group alignment calculation is performed at every node on the guide tree to generate a full alignment of all sequences.

Finally, a consensus sequence is derived from the multiple alignment. Alignment columns with more than 50% gaps are discarded, then the most likely base per column is determined, based on the substitution probabilities from last-train. The 50% gap threshold can be changed by the `-g` option.

5 Missing Sequences

lamassemble may show a warning message like this:

```
 lamassemble: using 29 out of 32 sequences (linked by pairwise
alignments)
```

This means that 3 out of 32 input sequences were discarded, because no alignments were found between them and the other reads. This is either because those 3 reads are genuinely unrelated to the others or because lamassemble (more specifically, LAST) failed to find some good alignments. As mentioned above under "How it works," the LAST step saves time by not finding all pairwise alignments between all reads: the aim is to link all reads indirectly, but this may fail. Such failure is not very common in our experience, and may not matter anyway. If it is a concern, the LAST step can be made more thorough-but-slow by increasing the m parameter:

```
 lamassemble -m100 last-train.mat sequences.fastq > consensus.
fasta
```

This sets the LAST rareness parameter m to 100 (presumably higher than the default), so it aligns each position in each read to a larger number of most-similar reads.

The default parameter settings are not catalogued in this chapter, because they may change in future, but they are shown by this command:

```
 lamassemble --help
```

6 Bad Alignments

In our experience, lamassemble usually works well, but sometimes it produces bad alignments and consensus sequences. A bad result tends to occur when the sequence has a near-exact tandem duplication that is large (comparable to the read length). In such cases, lamassemble may align wrong (paralogous) parts of reads.

One way to recognize a bad result is to align the consensus sequence to a reference genome, and visualize this alignment as a dotplot (*see* **Note 8**). A good result tends to have 45-degree lines (indicating straight alignment), and resembles the alignments of the raw reads: a bad result looks oddly different (Fig. 4).

There is no simple solution. Occasionally, the result improves upon increasing the z parameter:

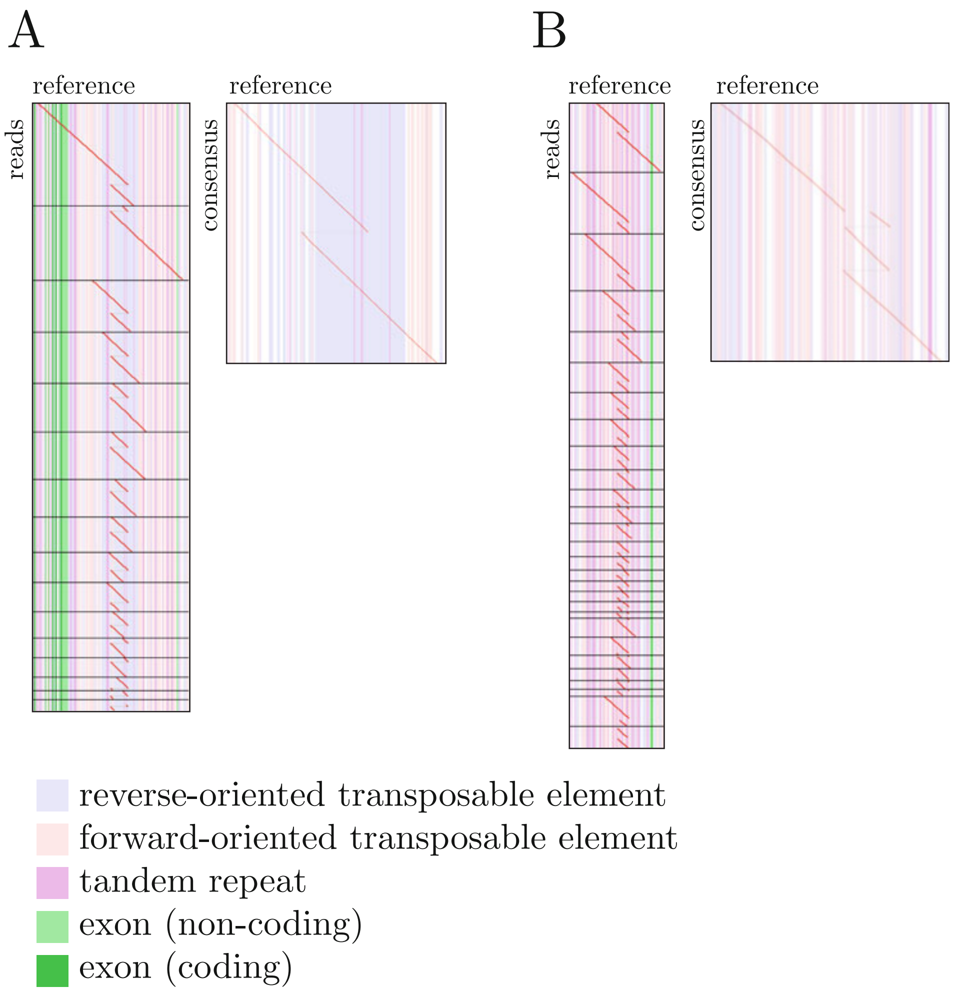


Fig. 4 Examples of good (**a**) and bad (**b**) lamassemble results. The "good" example shows alignments of 16 DNA reads (vertical) to the reference genome (horizontal), and alignment of the lamassemble consensus sequence to the genome. The horizontal black lines are boundaries between the 16 reads. The vertical stripes show annotations of the genome: green=exon, pink=forward-oriented transposable element, blue=reverse-oriented transposable element, purple=tandem repeat. The "bad" example shows 28 reads, and their lamassemble consensus sequence

```
 lamassemble -z999 last-train.mat sequences.fastq > consensus.
fasta
```

This parameter is the maximum gap length permitted in the LAST alignments: higher values increase run time. Actually, LAST imposes a hard upper bound on this parameter, which will be less than 999, so this setting really means "the maximum." Higher z may allow LAST to find a longer alignment spanning a large gap, which may change the guide tree and anchor priorities supplied to MAFFT.

7 Notes

1. It is harder to make accurate consensus sequences of short-period tandem repeats (STRs) than of non-repetitive sequence (e.g. Fig. 2). STR consensus sequences are less robust to small changes in the lamassemble parameters. So it may be worth maximizing z and increasing m. In Fig. 2, `-m100` and `-z100` options were used. We previously cut out the repetitive parts of reads with ±50bp flanks for global alignment [9], however, with lamassemble we recommend using either whole reads or longer flanks.
2. Sequences with highly unusual base composition require specialized last-train files. For example, genomes of *Plasmodium falciparum* or *Dictyostelium discoideum* are ~80% a+t, so it is important to use a last-train file that reflects this very-different composition.
3. last-train finds the rates of differences between reads and genome, which is a combination of sequencing errors and real differences between reads and genome. lamassemble assumes, however, that these are error rates, which is reasonable only if the rates of real differences are small relative to the error rates in the training reads. In order to apply lamassemble to reads with a high rate of real differences from a reference genome, it may be best to run last-train on a *different* set of reads, with few real differences, from the same sequencing technology.
4. Because the substitution error rates may not be strand-symmetric, it is best to be careful about which read strands are used. It is best to consistently use the original reads from the sequencer, not a mixture of original reads and reverse-complemented reads, for both last-train and lamassemble. (Consistently using reverse-complemented reads would work equally well.) lamassemble automatically flips strands if necessary to align the reads.
5. lamassemble assumes that the proportions of a:c:g:t in the molecules, before sequencing errors are introduced, are the same in each strand. In other words, it assumes that %a = %t and %c = %g in each strand (before sequencing errors). This assumption is violated by some kinds of data, e.g. bisulfite-converted DNA.
6. lamassemble converts u (uracil) to t (thymine), thus treats it just like t.
7. lamassemble allows ambiguous bases: m, r, w, s, y, k, v, h, d, b, n. For the most part, it treats them as unknown: it makes MAFFT use score 0 for aligning them to any base, and ignores them when determining a consensus base. (The LAST step

actually treats them non-neutrally, e.g. considering that w is a or t, as does MAFFT when run separately from lamassemble.)

8. Dotplots were drawn using last-dotplot (http://last.cbrc.jp/doc/last-dotplot.html) like this:

```
 last-dotplot --sort2=0 --strands2=1 --rot1=v --rot2=h --
labels1=3
  --rmsk1 rmsk.txt --genePred1 refFlat.txt aln.maf aln.png
```

The above command uses many options; the minimal command is

```
last-dotplot aln.maf aln.png
```

The main input is an alignment file `aln.maf`, and the output is an image file `aln.png`. The RepeatMasker annotation file `rmsk.txt` was obtained from the UCSC genome database (https://genome.ucsc.edu); it is also possible to use a `.out` file from http://www.repeatmasker.org. The gene annotation file `refFlat.txt` was obtained from the UCSC genome database.

References

1. Mitsuhashi S, Matsumoto N (2019) Long-read sequencing for rare human genetic diseases. J Hum Genet 65:1–9
2. Mongan AE, Tuda JSB, Runtuwene LR (2019) Portable sequencer in the fight against infectious disease. J Hum Genet 1–6. https://doi.org/10.1038/s10038-019-0675-4
3. Sakamoto Y, Sereewattanawoot S, Suzuki A (2019) A new era of long-read sequencing for cancer genomics. J Hum Genet 65:1–8
4. Stancu MC, Van Roosmalen MJ, Renkens I, Nieboer MM, Middelkamp S, De Ligt J, Pregno G, Giachino D, Mandrile G, Valle-Inclan JE, et al (2017) Mapping and phasing of structural variation in patient genomes using nanopore sequencing. Nat Commun 8 (1):1–13
5. Florian RT, Kraft F, Leitão E, Kaya S, Klebe S, Magnin E, vanRootselaar A-F, Buratti J, Kühnel T, Schröder C, et al (2019) Unstable TTTTA/TTTCA expansions in MARCH6 are associated with familial adult myoclonic epilepsy type 3. Nat Commun 10(1):1–14
6. Corbett MA, Kroes T, Veneziano L, Bennett MF, Florian R, Schneider AL, Coppola A, Licchetta L, Franceschetti S, Suppa A, et al (2019) Intronic ATTTC repeat expansions in STARD7 in familial adult myoclonic epilepsy linked to chromosome 2. Nat Commun 10 (1):1–10
7. Yeetong P, Pongpanich M, Srichomthong C, Assawapitaksakul A, Shotelersuk V, Tantirukdham N, Chunharas C, Suphapeetiporn K, Shotelersuk V (2019) TTTCA repeat insertions in an intron of YEATS2 in benign adult familial myoclonic epilepsy type 4. Brain 142(11):3360–3366
8. Ishiura H, Shibata S, Yoshimura J, Suzuki Y, Qu W, Doi K, Almansour MA, Kikuchi JK, Taira M, Mitsui J, et al (2019) Noncoding CGG repeat expansions in neuronal intranuclear inclusion disease, oculopharyngodistal myopathy and an overlapping disease. Nat Genet 51(8):1222–1232
9. Sone J, Mitsuhashi S, Fujita A, Mizuguchi T, Hamanaka K, Mori K Koike H, Hashiguchi A, Takashima H, Sugiyama H, et al (2019) Long-read sequencing identifies GGC repeat expansions in NOTCH2NLC associated with neuronal intranuclear inclusion disease. Nat Genet 51(8):1215–1221
10. A pipeline for complete characterization of complex germline rearrangements from long DNA reads. Genome Med 12, 67 (2020).

11. De Coster W, De Rijk P, De Roeck A, De Pooter T, D'Hert S, Strazisar M, Sleegers K, Van Broeckhoven C (2019) Structural variants identified by Oxford Nanopore PromethION sequencing of the human genome. Genome Res 29(7):1178–1187
12. Hamada M, Ono Y, Asai K, Frith MC (2017) Training alignment parameters for arbitrary sequencers with LAST-TRAIN. Bioinformatics 33(6):926–928
13. Mitsuhashi S, Frith MC, Mizuguchi T, Miyatake S, Toyota T, Adachi H, Oma Y, Kino Y, Mitsuhashi H, Matsumoto N (2019) Tandem-genotypes: robust detection of tandem repeat expansions from long DNA reads. Genome Biol 20(1):58
14. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge

Chapter 10

Automated Removal of Non-homologous Sequence Stretches with PREQUAL

Iker Irisarri, Fabien Burki, and Simon Whelan

Abstract

Large-scale multigene datasets used in phylogenomics and comparative genomics often contain sequence errors inherited from source genomes and transcriptomes. These errors typically manifest as stretches of non-homologous characters and derive from sequencing, assembly, and/or annotation errors. The lack of automatic tools to detect and remove sequence errors leads to the propagation of these errors in large-scale datasets. PREQUAL is a command line tool that identifies and masks regions with non-homologous adjacent characters in sets of unaligned homologous sequences. PREQUAL uses a full probabilistic approach based on pair hidden Markov models. On the front end, PREQUAL is user-friendly and simple to use while also allowing full customization to adjust filtering sensitivity. It is primarily aimed at amino acid sequences but can handle protein-coding nucleotide sequences. PREQUAL is computationally efficient and shows high sensitivity and accuracy. In this chapter, we briefly introduce the motivation for PREQUAL and its underlying methodology, followed by a description of basic and advanced usage, and conclude with some notes and recommendations. PREQUAL fills an important gap in the current bioinformatics tool kit for phylogenomics, contributing toward increased accuracy and reproducibility in future studies.

Key words Filtering, Genomics, HMM, Homology, Phylogenomics, Sequence analysis

1 Introduction

The assembly of large datasets used in phylogenomic and comparative genomic analyses heavily relies on automatic tools, typically run sequentially into a pipeline. There are, however, few automatic tools for quality control, resulting in these important controls being performed (semi-) manually or worse, ignored. One such control is the removal of primary sequence errors inherited from source genomes and transcriptomes. These errors typically arise from sequencing or assembly errors that insert (or delete) a nucleotide, leading to the disruption of the reading frame in translated protein-coding genes. With poor sequence quality, it is not uncommon that the reading frame is re-established by a second downstream deletion (or insertion), producing stretches of non-homologous amino acids within an otherwise legitimate sequence. Another common situation is represented by poorly annotated eukaryotic genes where intron-exon boundaries are incorrectly predicted, resulting in translated intronic regions being forced into multiple sequence alignments (MSAs).

The impact of these sequence errors in downstream analyses is not yet fully understood, but it is reasonable to assume that they will adversely affect phylogenetic analyses. Persistence of non-homologous sequence stretches will negatively affect the quality of MSA, which is known to impact phylogenetic accuracy [1]. The non-homologous sequence itself might also cause problems. If the phylogenetic signal in the source data is weak, then errors induced by the primary sequence error or its effect on the MSA might lead to false groupings in the tree, akin to the systematic errors caused by model misspecifications [2, 3]. Furthermore, primary sequence errors can bias the estimation of natural selection coefficients (dN/dS) in a similar manner to errors in MSA, because they lead to an artificially inflated number of non-synonymous substitutions [4]. Di Franco et al. [5] have found that segment filtering leads to improved branch length estimation and a lower false-positive rate in detecting positive selection.

When we look at MSAs, primary sequence errors are typically evident as stretches of residues that share no sitewise homology with other sequences in otherwise evolutionarily conserved regions (*see* Fig. 1a). These stretches might be identified in a labor-intensive and qualitative way through looking at MSAs with homologous sequences from other species, but fixing this problem is at best painstaking for a phylogenomic study with hundreds of genes and species. For this reason, this step has been ignored by most phylogenomic studies so far. Even after that hard work, the result is based on individual opinion, making it subjective and not reproducible.

To address this problem we developed PREQUAL (PRE-alignment QUALity filter; [6]): a command line tool that, given a set of unaligned homologous sequences as input, can identify and mask regions with non-homologous adjacent characters. Our approach (*see* Fig. 1b) is different from the standard trimming of MSAs to remove poorly aligned columns (*see* Fig. 1c), for which several tools exist, including BMGE [7], trimAl [8], Gblocks [9], and Divvier [10]. We deal with non-homologous residues in individual sequences that are not necessarily associated with regions of poor alignment. In fact, PREQUAL uses unaligned sets of sequences and thus does not assume any fixed sequence alignment. Instead of simply removing these errors, PREQUAL masks them in order to maintain the original distance between residues, thereby facilitating the inference of positional homology in the MSA step.

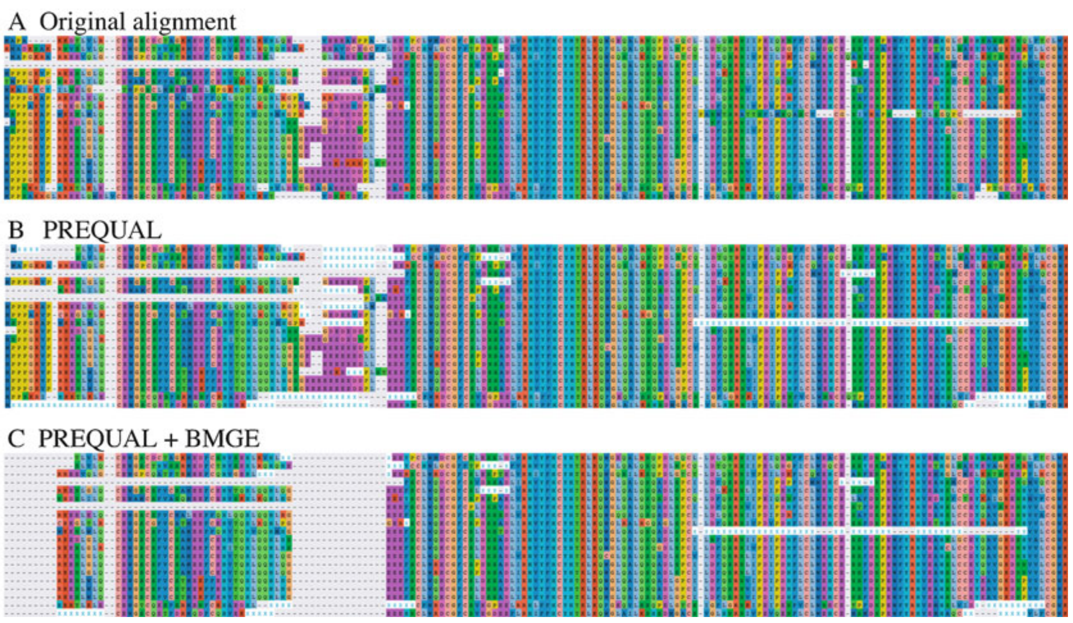


Fig. 1 Primary sequence errors and performance of stretch-masking vs. block filtering. (**a**) Alignment of Hedgehog-interacted protein (HHIP) orthologs from vertebrate genomes (OrthoDB acc. EOG090702V5; http:/www.orthodb.org/; accessed 07/11/2017). (**b**) PREQUAL masks non-homologous residue stretches in individual sequences with X (light blue). Note that PREQUAL works on unaligned sequences but here aligned for easy identification of masked residues. (**c**) Alignment block filtering with BMGE on PREQUAL-treated and aligned sequences. Images produced with AliView [20]

Moreover, masking likely non-homologous stretches should facilitate the inference of MSAs, thereby improving MSA quality and reducing the known biases in downstream phylogenomic and comparative genomic analyses.

2 Methods

2.1 Methodological Intuition

Before discussing the technical details of what PREQUAL does, we will give a broad overview of what it is trying to achieve and how it works. We can formulate the problem of non-homologous residues as a hypothesis for each individual residue in every sequence stating "is there evidence that this residue shares sitewise homology with another residue in my data set?" In other words, if I were to consider all the probable ways by which a residue could have evolved, is there support that this residue shares ancestry with any other residue in my data? If the answer is yes, then that residue should be carried forward to future steps (although we note it could be removed later for other reasons, such as poor alignment quality). If the answer is no, then that residue should be excluded from future steps, either through masking to maintain the relative positions of the residues or through simple removal.

With this hypothesis in mind, the next question is how to quantify the support for homology and pick a threshold that specifies yes or no. Our quantification works by comparing a sequence with every other sequence in our data, and for each residue in that sequence, we calculate the maximal probability of that residue sharing sitewise homology with any other residue in the data using an explicit probability model. This gives a score for every residue in every sequence, which is compared to an a priori chosen threshold to decide whether to keep or remove the residue. In order to derive an efficient threshold, we took both a simulation approach—where we introduced errors to error-free simulated sequences and found the best score to identify those errors—and an empirical approach looking at real data and assessing whether the removed residues showed signs of errors. Our results showed that both approaches yielded very similar thresholds, giving us confidence that PREQUAL works as intended [6].

This intuition is also helpful for explaining why PREQUAL is designed to work for amino acid sequences (or translated to amino acids from protein-coding DNA), but not straight on DNA sequences. In order to test our hypothesis of sitewise homology, we need sequences to be relatively complex, so that we can reliably distinguish true homologies from errors. Amino acid sequences have 20 possible characters and are therefore much more complex than DNA sequences that have only four. To put this in numbers, consider a sequence of length 10: there are around one million possible DNA sequences ($4^{10} = 1{,}048{,}576$) but around 10 trillion possible amino acid sequences ($20^{10} = 10.24 \times 10^{-12}$). The vastly greater space of possible amino acid sequences makes identifying similarities—and therefore errors—much easier.

2.2 Statistical Approach

PREQUAL uses a full probabilistic approach based on pair hidden Markov models (pairHMM) to describe the relationship between sequence pairs. Given a parameterized pairHMM, it calculates the posterior probability (PP) of a character being related to a character from another sequence and filters out (masks) characters with insufficient evidence of homology. PairHMMs consist of three hidden states defining the relationship between two sequences X and Y: a match state that describes shared ancestry between X and Y through a process of substitution and insert and delete states that describe gain and loss in sequence X, respectively. Given a parameterized pairHMM, it is possible to calculate the PP of a character from X being related to a character from Y using the forward-backward algorithm [11]. In PREQUAL, a slightly different PP is calculated to capture the PP of a character in sequence $X = \{x_i\}$ sharing a common ancestor with any character in the set of n sequences being considered $\mathbf{A} = \{Y^1, \ldots, Y^n\}$, where $X \in \mathbf{A}$; the set of sequences without X is $\mathbf{A}' = \mathbf{A} - X$, and a pairHMM can be run on the pair (X, Y) to obtain the posterior probability of $\Pr(x_i, y_j)$ using forward-backward. First, we can calculate the maximal PP of x_i sharing a common ancestor with any character in Y as $\Pr(x_i, Y_*) = \max_{y_j \epsilon Y}\left\{\Pr\left(x_i, y_j\right)\right\}$. Then we want to find the maximal PP of x_i sharing a common ancestor with any of the other sequences: $\Pr(x_i|\text{Ancestor}) = \max_{Y \epsilon \mathbf{A}'}\{\Pr(x_i, Y_*)\}$. The value of $\Pr(x_i|\text{Ancestor})$ can be computed for every character of every sequence in $\mathbf{A}$, and an appropriate threshold τ can be used as a cutoff to discriminate between characters with enough evidence for shared ancestry that should be carried through to the alignment phase and those that should be discarded (masked).

In practice, PREQUAL calculates PPs using a bounded pairHMM with a substantially modified version of Zorro [12]. It uses a heuristic approach for calculating $\Pr(x_i|\text{Ancestor})$ by choosing a set of sequences from $\mathbf{S}'$ based on evolutionary divergence [13] and sequence coverage to reduce the number of pairHMM calculations that need to be performed. This heuristic samples only a subset of sequences to find the $\max(p(x_i|\text{Ancestor}))$ ensuring that the closest sequences are included and they have adequate similarity over enough of the characters. The default is to examine the ten closest sequences ensuring that at least three of them are equal to or longer than the median length of all the sequences. PREQUAL also avoids computing the entire dynamic programming matrix for the pairHMM where possible, by bounding the distance any path can take through that matrix from the diagonal.

The residue-specific PP is the main filter in PREQUAL. The default threshold is PP = 0.994, which has been derived from a ROC curve so that ≥95% of correct amino acids are retained while removing >90% of the errors (*see* Fig. 4). The performance of the default threshold has also been validated on various real phylogenomic datasets (*see* Subheading 4).

3 Program Usage

PREQUAL is designed to be as simple and lightweight to use as possible. The default values should work well for most phylogenomic datasets. Nevertheless, PREQUAL provides the user with many options for customizing its behavior and comes with a detailed manual.

3.1 Download and Installation

The program PREQUAL is written in C/C++, and it is available through a GNU GPL v3.0 from Simon Whelan's GitHub (https://github.com/simonwhelan/prequal).

PREQUAL can be built from source with a simple "make" command in Linux and OS X. After installation, add it to your path if you want PREQUAL to be accessible from anywhere in your system.

3.2 Basic Usage

PREQUAL runs on the command line. The majority of users will find the basic usage sufficient (obtained by typing the program name without any argument). Given an input file containing a set of homologous sequences (in FASTA format) and assuming that PREQUAL is in the current directory, type:

```
./prequal my_seq.fasta
```

This command will run PREQUAL on the input file. The input file should be a set of homologous sequences in FASTA format and must not contain stop codons (*).

The default output of PREQUAL consists of at least two files:

`my_seq.fasta.filtered`: main output containing the sequences after filtering.

`my_seq.fasta.PP`: internal file containing the calculated PP. The interest of saving the PPs is to rerun PREQUAL with different settings on the same data without the need of recalculating PPs, the most computationally intensive step (*see* **Note 1**).

`my_seq.fasta.warning`: this file might be also generated, containing warnings when too much of a sequence (or all of it) has been filtered out.

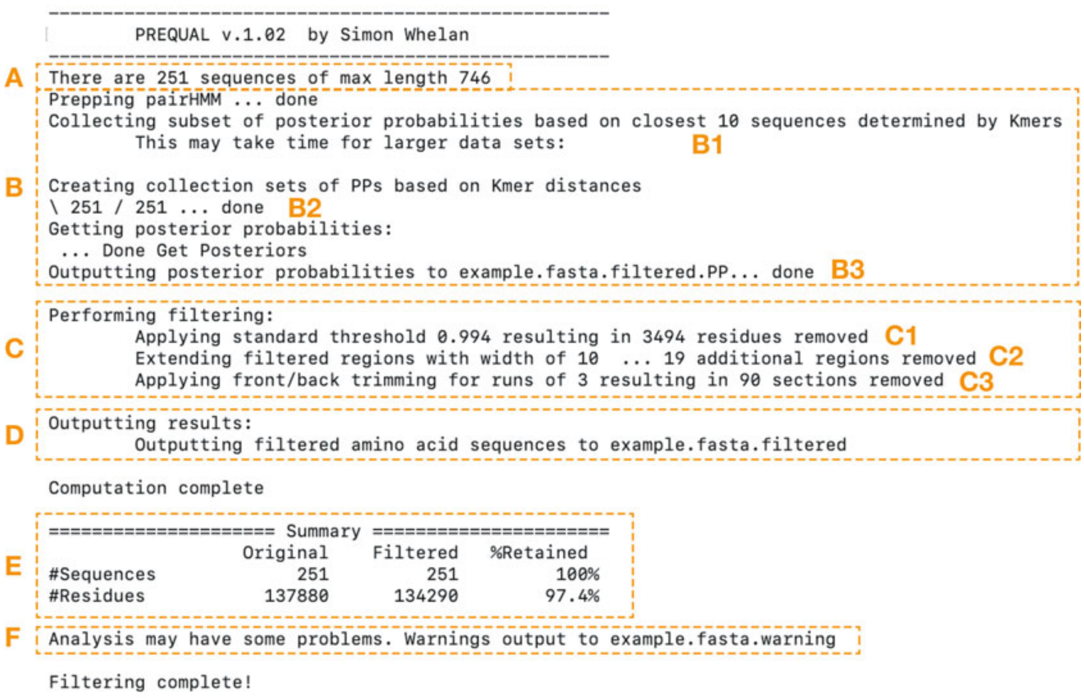
Figure 2 shows a typical output printed to the screen. It can be separated into the following sections:

A Summary of input data.

B Information about the computation of PPs. By default, PREQUAL's heuristic will compare each sequence to the ten most similar sequences given adequate coverage (B1). Progress spinners provide an indication of the progress (B2). A confirmation will be printed when PPs are calculated (B3).

C This section refers to the actual filtering step. It provides information on the applied PP threshold and the number of residues that fall below it and thus will be removed or masked (C1). Often, residues with low PP form stretches, and these might be joined by PREQUAL and this information is also printed (C2). By default, PREQUAL fully removes low PP residues in the N- and C-termini, whereas residues in the "core" region are masked with a "X" character (C3).

D Name of main output file.



```
---------------------------------------------------
        PREQUAL v.1.02  by Simon Whelan
---------------------------------------------------
There are 251 sequences of max length 746
Prepping pairHMM ... done
Collecting subset of posterior probabilities based on closest 10 sequences determined by Kmers
        This may take time for larger data sets:

Creating collection sets of PPs based on Kmer distances
\ 251 / 251 ... done
Getting posterior probabilities:
 ... Done Get Posteriors
Outputting posterior probabilities to example.fasta.filtered.PP... done

Performing filtering:
        Applying standard threshold 0.994 resulting in 3494 residues removed
        Extending filtered regions with width of 10  ... 19 additional regions removed
        Applying front/back trimming for runs of 3 resulting in 90 sections removed

Outputting results:
        Outputting filtered amino acid sequences to example.fasta.filtered

Computation complete

==================== Summary =====================
                  Original    Filtered    %Retained
#Sequences             251         251         100%
#Residues           137880      134290        97.4%

Analysis may have some problems. Warnings output to example.fasta.warning

Filtering complete!
```

Fig. 2 Typical screen output of PREQUAL, showing basic information of input dataset, analysis, and results. See main text for detailed description

E Summary of the analysis showing the number of sequences and residues in the input ("Original") and output ("Filtered"). Note that individual sequences might be entirely removed, in which case a message will be printed to ".warning."

F Sometimes, PREQUAL will generate an additional warning file. This information is important and should be inspected. Examples of possible warnings might be "WARNING: 86.94% of sequence removed/missing for [5] Seq1" or "WARNING: Fully removed sequence [5] Seq2."

3.3 Using PREQUAL with DNA Sequences

When the input data are nucleotide sequences of protein-coding genes, these are first translated in silico upon automatic selection of the appropriate genetic code (this can also be provided by the user), sequences are then filtered at the amino acid level as described before, and the corresponding filtered DNA sequences are printed to the output.

A simple run in PREQUAL with DNA sequences might look like this:

```
./prequal my_seq_DNA.fasta
```

The program will first confirm that the input consists of DNA sequences by writing to the screen: "Found only DNA sequences. Doing translations."

In this case, the default consists of at least four files:

`my_seq_DNA.fasta.dna.filtered`: main output containing the masked nucleotide sequences.

`my_seq_DNA.fasta.filtered`: corresponding masked amino acid sequences.

`my_seq_DNA.fasta.translation`: information regarding the translation. From nucleotide to amino acid sequences and the applied genetic code.

`my_seq_DNA.fasta.PP`: internal file containing the calculated PP.

`my_seq_DNA.fasta.warning`: an additional warning file will most likely be generated, containing important information about the sequences.

While using PREQUAL with nucleotide sequences of protein-coding genes is possible (and highly recommended), some restrictions apply. All sequences must be in the first reading frame and must have all its codons complete, and sequences must not contain stop codons (`*`). In case of any ambiguous nucleotide (not A, C, G, or T), the entire codon will be ignored and treated as gaps in the output. If PREQUAL detects that the input file contains DNA sequences, it will try to automatically select the right genetic code to translate the DNA sequences. This automatic code selection is heavily guided by stop codons, therefore the need to ensure that no internal stop codons are present in the input. Small differences in the genetic code when doing the translations should not strongly affect the filtering step, but it is highly recommended that the user checks the automatically selected genetic code, specified in the translation file. It is also possible to enforce the use of the universal genetic code using the option "`-forceuniversal`" (see below). When using PREQUAL with DNA data, it is highly recommended to check the warning and translation files to ensure that the filtering has been performed adequately.

3.4 Advanced Usage

In the following, we describe all available settings for fine-tuning the behavior of PREQUAL and adapt it to the user's preferences. A short description of all available options can be obtained in the command line by typing "`--h all`." (*see* also **Notes 1** and **2**).

3.4.1 Options Affecting the Definition of Core Regions and Filtering

To account for the different evolutionary patterns observed across most proteins, PREQUAL defines core and flanking regions. A core region is defined as the central part of the protein that is evolutionarily relatively well conserved, and its flanking regions, attributable to N- and C-termini, are allowed to be less conserved. In practice, core regions are defined by the presence of at least three contiguous residues with high PP. By default, residues with low PP within the core region are masked with an "X" whereas those in the flanking (noncore) regions are simply removed.

The default behavior regarding the definition of core regions and filtering can be controlled with the following options:

`-corerun X`: Defines the number (X) of contiguous residues with high PP that are used to define a core region. Low values of X will make the program more generous at defining the core, whereas high values will make the program more conservative. Default is 3.

`-nocore`: No core region is defined. Default is to define a core region, which is defined by the presence of at least three consecutive high-PP residues.

`-removeall`: Removes all low PP residues rather than only those in the flanking regions. This option should be used with caution. The masking of residues within the core region maintains the original distance between residues, thereby facilitating the inference of the original positional homologies. The complete removal of residues can negatively affect MSA.

`-corefilter X`: Defines the character used to mask residues in the core region. Default is "X" and alternatives can only be a single character. Alternative characters might help in the visualization.

`-noremoverepeat`: By default, PREQUAL will attempt to remove long identical repeats that can occur within individual sequences, most often due to sequencing or assembly errors. By selecting this option, repeats will not be removed (*see* also **Note 3**).

3.4.2 Options Related to DNA Sequences

The following options affect how PREQUAL deals with protein-coding DNA sequences:

`-nodna`: By default, PREQUAL will try to guess whether the input data are amino acids or nucleotides. Using this option forces PREQUAL to read input as amino acid sequences. This might be useful for proteins extremely enriched in alanine (A), cysteine (C), glycine (G), and/ or threonine (T).

`-forceuniversal`: This option forces PREQUAL to always use the universal code. By default PREQUAL will attempt to choose the right genetic code for translating the DNA sequences by detecting codons that differ from the standard code.

3.4.3 Options Affecting Output Formats

The output files and naming conventions can be modified using the following settings:

`-outsuffix X`: By default PREQUAL prints out filtered data to the file ".filtered" using the input file as a base name. This option will change the suffix of the file containing the filtered sequences.

`-dosummary`: This will generate a new output file ".summary" containing a summary of the analyses, including information on the proportion of residues removed per sequence (in total and within core regions) and some helpful cutoffs of the proportion of removed data with their associated PP thresholds, which can be informative to select a custom cutoff.

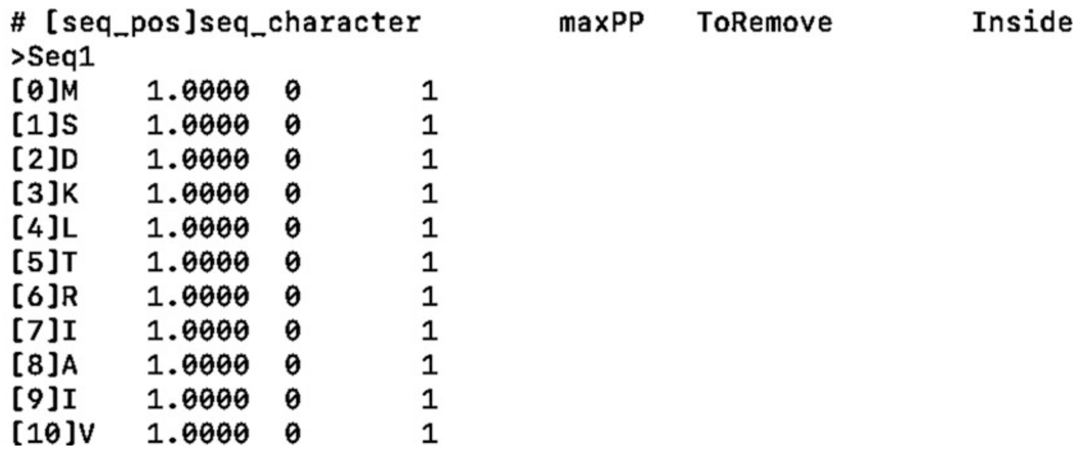
`-dodetail`: This will generate a new output file ".detail" where, for each sequence and residue, the following information will be printed out: estimated PP (maxPP; range 0,1), whether the residue has been filtered out (toRemove; 0 = FALSE; 1 = TRUE), and whether it is within the core region (Inside; 0 = FALSE; 1 = TRUE) (*see* Fig. 3). This file can be used to inspect the PP values of specific residues to personalize the filtering options.

`-noPP`: Do not output the posterior probability file (".PP"). Note that this will require to recalculate PPs at every run of PREQUAL (*see* also **Note 1**).

3.4.4 Options Affecting Posterior Probabilities and Filtering

The following options might be used to modify the residue filtering as well as the heuristics used for the calculation of PPs:

`-filterthresh X`: Modifies the PP threshold. This is the main filter in PREQUAL, and any residue with a PP below this



```
# [seq_pos]seq_character          maxPP    ToRemove          Inside
>Seq1
[0]M     1.0000   0        1
[1]S     1.0000   0        1
[2]D     1.0000   0        1
[3]K     1.0000   0        1
[4]L     1.0000   0        1
[5]T     1.0000   0        1
[6]R     1.0000   0        1
[7]I     1.0000   0        1
[8]A     1.0000   0        1
[9]I     1.0000   0        1
[10]V    1.0000   0        1
```

Fig. 3 Example of a detailed output file generated with "`-dodetail`" option. See main text for detailed description

threshold will be filtered out. The default value (X = 0.994) has been defined based on analyses of simulated data and validated on a wide range of real sequence datasets (*see* Subheading 4). In principle, filterthresh can take any value between 0 and 1, corresponding to situations where all residues would be kept or filtered out, respectively.

`-filterprop X`: An alternative way of filtering the data is by defining the proportion of the original data (X%) that the user is willing to loose. When using this option, PREQUAL will automatically adjust the PP threshold accordingly. In practice, PREQUAL will often filter out a slightly higher proportion of data than specified because of the way regions of low confidence are joined together and how N- and C-termini are dealt with. This option will also print out to screen several cutoffs of the proportion of removed data with their associated PP thresholds, which might be informative to select the desired cutoff. The options "`-filterprop`" and "`-filterthresh`" may not be used together.

`-pptype closest/longest/all [Y]`: This setting modifies the algorithm that chooses the subset of sequences that will be used to calculate PPs for each individual sequence. The default is to compare the ten closest sequences defined by Kmer distances ("`--pptype closest`"). The number of closest relatives considered might be raised to improve the accuracy of the PPs by specifying Y (e.g., "`--pptype closest 20`"). It is also possible to specify that PPs are calculated with the ten longest sequences ("`-pptype longest`"; the number of longest sequences may also be changed with Y). We recommend to use the latter option with caution because the longest sequences might contain the most errors. A last possibility is to consider all sequences ("`-pptype all`"). The last option might be very slow, particularly for datasets with many sequences.

`-filterjoin X`: The default behavior of PREQUAL is to join low PP residues into masked stretches if these are separated by fewer than ten residues. This option allows modifying the number of residues required between two low PP residues (X) so that the entire sequence stretch is filtered out. Low values of X are more generous and will keep more data, whereas high values of X are more stringent and will filter out more data.

`-nofilterlist X`: This option allows to provide a file "X" containing a list of full sequence names (i.e., corresponding to the FASTA headers) to be ignored by PREQUAL during the masking step. This option might be useful if some sequences in the input file are expected to be highly divergent a priori, which might lead to incorrectly filtering out too much legitimate data (*see* **Note 4**).

`-nofilterword X`: Similar to the above option, it is possible to define a file "X" that contains a list of key words that might appear within sequence names (i.e., within the FASTA headers) and these will not be masked by PREQUAL.

4 Benchmarking

The accuracy and efficiency of PREQUAL in detecting true errors were tested using simulated error-free sequences, which were corrupted by including two types of errors at known—but random—positions, and tracking the number of errors (true and false) that were detected by PREQUAL [6]. Briefly, sets of 100 gene alignments were simulated in INDELible v.1.03 [14] using the WAG+Γ model ($\alpha = 0.5$ or 1.8) on a published tree containing both long and short terminal branches [15]. Simulated alignments were 500 amino acids long and contained a core region (450 amino acids) flanked by more gappy regions (25 amino acids each) aiming to mimic typical exon alignments. Three sets of 100 alignments were simulated, each differing on the level of gappyness: low (0.01 gap rate for the core and 0.02 for flanking regions), medium (0.02 and 0.05), and high (0.1 and 0.2). Individual alignment files were then corrupted by randomly inserting errors in sequences (proportional to sequence length) and locations. The errors corresponded to random amino acids drawn proportionally from the residue frequencies of the WAG model. Errors were either inserted at random positions, mimicking misannotation errors derived from wrong gene models, or replaced parts of the original sequences, mimicking frameshifts produced by indels at the nucleotide level. Three error rates were tested: low (0.001 errors per amino acid, ~22 errors per file), medium (0.002: ~44 errors), and high (0.003: ~66 errors). In addition, different lengths of individual errors were also tested: 10, 20, and 30 amino acids for their final expected lengths (the actual lengths were drawn from a geometric distribution). Overall, a total of 108 experimental conditions were simulated including three sets of alignments of varying gappyness, three error rates, three error lengths, and two types of errors (misannotations and frameshifts). Simulated gene files were analyzed with PREQUAL v.1.0 and the results are summarized in Table 1.

Overall, PREQUAL performed well under all tested conditions but was particularly efficient at detecting errors inserted at random positions mimicking misannotations (>98% were captured). Accuracy (proportion of correctly identified true and erroneous residues) was in most cases >90%, although it lowered up to 73% when very gappy alignments were considered. Regarding frameshift errors that replaced parts of the original sequences, PREQUAL generally detected >93% of erroneous residues, except for short

Table 1
Performance of PREQUAL on simulated data

		Misannotations			Frameshifts		
		No. of characters	Accuracy (%)	Errors captured (%)	No. of characters	Accuracy (%)	Errors captured (%)
Gappyness	Low	2.339.354	95.47	99.67	2.250.400	95.60	93.21
	Mid	2.345.437	91.84	99.68	2.255.919	91.89	94.60
	High	2.362.243	73.10	99.76	2.272.380	73.14	96.61
Number of errors	Low	2.300.164	92.26	99.81	2.255.919	92.36	94.24
	Mid	2.390.366	91.47	99.78	2.255.919	91.49	94.62
	High	2.390.366	91.47	99.78	2.255.919	91.49	94.62
Expected error length	10 AA	2.296.403	91.77	99.16	2.255.919	91.90	68.97
	20 AA	2.390.366	91.47	99.78	2.255.919	91.49	94.62
	30 AA	2.389.967	91.99	99.87	2.255.919	91.98	98.15

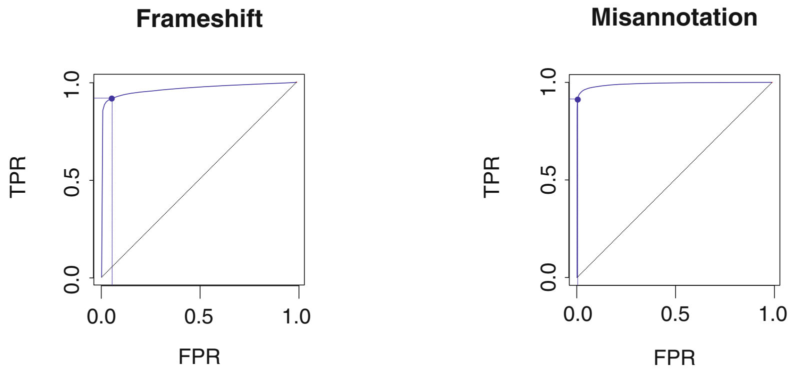


Fig. 4 ROC curve showing the relationship between true-positive rates (TPR) and false-positive rates (FPR) for frameshift and misannotation errors based on simulated data. Dots on curves mark the critical point where ≥95% of correct amino acids are retained and >90% of the errors are removed, corresponding to the default PP threshold of 0.994

error stretches of ~10 residues where 69% of errors were captured. The accuracy of frameshift detection followed the same trend as for misannotations. The proportion of legitimate residues removed by PREQUAL on error-free sequences was ~7% for most tested conditions.

Using a ROC curve, we derived the default PP threshold of 0.994 so that ≥95% of correct amino acids were retained while removing >90% of frameshift and misannotation errors (*see* Fig. 4). The area under the curve (AUC), representing the performance of PREQUAL as classifier, was 0.965 and 0.992 for frameshifts and misannotations, respectively. The default threshold derived from simulated data was further evaluated on various real phylogenomic datasets [3, 16, 17]. A careful inspection of the results suggested that the overwhelming majority of errors identifiable by eye were removed using the default PP threshold, showing its appropriateness for a wide range of datasets characterized by very different levels of sequence divergence, from ca. 20 to 2000 million years ago.

PREQUAL can handle typical datasets on a common laptop computer. For example, the analysis of 107 sequences with a maximal length of 675 amino acids took 42 s on a computer equipped with a 2.7 Ghz i7 processor, whereas a larger set of 272 sequences with a maximal length of 1149 amino acids took 139 s (*see* also **Note 5**).

5 Notes

1. Although the default settings should work well for most applications, it is possible to fine-tune PREQUAL's behavior to specific datasets and user needs. To facilitate this process, PP values (the computationally most expensive part) are stored into an output file (".PP") by default. When a PP file with the same root name as the input file is present in the same directory, PREQUAL will get the PP values from that file and only perform the filtering step, which is almost immediate. This functionality allows to experiment with different filtering options without additional computational cost.
2. PREQUAL is not designed to detect highly divergent paralogs. If a single divergent paralog exists per input file, PREQUAL might be able to filter it out if the evidence for homology (PP) is low. However, this will be almost impossible if two or more paralogs are present because the evidence of homology between them might be high and thus paralogs will be retained. Similarly, PREQUAL will have problems to remove primary sequence errors if these happen at similar relative locations in two or more sequences. Even though most errors are expected to not have this distribution, it can happen if, e.g., sequences contain primer sequences or they mistakenly incorporate the same intron due to a propagated annotation error. These situations violate the underlying assumptions of PREQUAL. In the presence of multiple divergent paralogs or errors at similar locations, PREQUAL will assume that the observed high similarity can only arise due to shared ancestry and therefore the affected residues will most likely be kept. Therefore, it is recommended to check the filtered data and to deal with paralogs with more appropriate methods.

3. The use of PREQUAL on repetitive sequences is not recommended. PREQUAL was designed to work with simple protein-coding homologous sequences. If input sequences contain homologous regions within the sequence itself, such as repeated domains or other tandem repeats, establishing the homology between the repeats becomes challenging. If dealing with such cases, it is recommended to carefully check the results and decide whether PREQUAL adequately dealt with the data at hand. Sometimes, repeated regions might occur within individual sequences by error (e.g., erroneous gene models). By default, PREQUAL will assume that long identical repeats are due to errors and it will try to remove them (printing a note to the warning file). This can be avoided using the "`-noremoverepeat`" option.
4. PREQUAL will have trouble to handle highly divergent sequences in a set of otherwise conserved proteins. This can be the case for sequences from parasites or prokaryotic homologs analyzed together with eukaryote sequences. In these cases, PREQUAL will likely mask many error-free—but divergent—residues. In order to overcome this problem, it is possible to provide a list of taxa that should be ignored during homology inference (*see* Subheading 3.4.4).
5. PREQUAL is computationally light and runs on a single CPU. With genome-scale data, it is routine to analyze hundreds or thousands of input files. Depending on the number and size of the input files, this might be done with a "for" loop in Bash. It is also possible to take advantage of parallelization on multicore systems, e.g., using GNU parallel [18] or incorporate PREQUAL into pipelines or workflow managers such as Snakemake [19].

Acknowledgments

We would like to thank Kazutaka Katoh for the possibility of contributing this chapter. Max E. Schön provided comments on an earlier version. II acknowledges the support from a Juan de la Cierva-Incorporación postdoctoral fellowship (IJCI-2016-29566) from the Spanish Ministry of Science and Competitiveness (MINECO). This work in the lab of FB is supported by a fellowship from Science for Life Laboratory. SW thanks the Carl Tryggers Stiftelse and Uppsala University for support.

References

1. Chatzou M, Floden EW, Di Tommaso P, Gascuel O, Notredame C (2018) Generalized bootstrap supports for phylogenetic analyses of protein sequences incorporating alignment uncertainty. Syst Biol 67(6):997–1009
2. Philippe H, Brinkmann H, Lavrov DV, Littlewood DTJ, Manuel M, Wörheide G et al (2011) Resolving difficult phylogenetic questions: why more sequences are not enough. PLoS Biol 9(3):e1000602
3. Irisarri I, Meyer A (2016) The identification of the closest living relative(s) of tetrapods: phylogenomic lessons for resolving short ancient internodes. Syst Biol 65(6):1057–1075
4. Schneider A, Souvorov A, Sabath N, Landan G, Gonnet GH, Graur D (2009) Estimates of positive Darwinian selection are inflated by errors in sequencing, annotation, and alignment. Genome Biol Evol 1:114–118
5. Di Franco A, Poujol R, Baurain D, Philippe H (2019) Evaluating the usefulness of alignment filtering methods to reduce the impact of errors on evolutionary inferences. BMC Evol Biol 19 (1):21
6. Whelan S, Irisarri I, Burki F (2018) PREQUAL: detecting non-homologous characters in sets of unaligned homologous sequences. Bioinformatics 34(22):3929–3930
7. Criscuolo A, Gribaldo S (2010) BMGE (Block Mapping and Gathering with Entropy): a new software for selection of phylogenetic informative regions from multiple sequence alignments. BMC Evol Biol 10(1):210
8. Capella-Gutiérrez S, Silla-Martínez JM, Gabaldón T (2009) trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. Bioinformatics 25(15):1972–1973
9. Castresana J (2000) Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Mol Biol Evol 17 (4):540–552
10. Ali RH, Bogusz M, Whelan S (2019) Identifying clusters of high confidence homologies in multiple sequence alignments. Mol Biol Evol 36(10):2340–2351
11. Durbin R, Eddy SR, Krogh A, Mitchison GJ (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
12. Wu M, Chatterji S, Eisen JA (2012) Accounting for alignment uncertainty in phylogenomics. PLoS One 7(1):e30288
13. Bogusz M, Whelan S (2017) Phylogenetic tree estimation with and without alignment: new distance methods and benchmarking. Syst Biol 66(2):218–231
14. Fletcher W, Yang Z (2009) INDELible: a flexible simulator of biological sequence evolution. Mol Biol Evol 26(8):1879–1888
15. Whelan NV, Kocot KM, Moroz TP, Mukherjee K, Williams P, Paulay G et al (2017) Ctenophore relationships and their placement as the sister group to all other animals. Nat Ecol Evol 1(11):1737–1746
16. MacLeod A, Irisarri I, Vences M, Steinfartz S (2015) The complete mitochondrial genomes of the Galápagos iguanas, *Amblyrhynchus cristatus* and *Conolophus subcristatus*. Mitochondr DNA Part A 27(5):3699–3700
17. Burki F, Kaplan M, Tikhonenkov DV, Zlatogursky V, Minh BQ, Radaykina LV et al (2016) Untangling the early diversification of eukaryotes: a phylogenomic study of the evolutionary origins of Centrohelida, Haptophyta and Cryptista. Proc R Soc B-Biol Sci 283 (1823):20152802
18. Tange O (2015) GNU Parallel 20150322 ('Hellwig'). USENIX Magazine 36:42–47
19. Köster J, Rahmann S (2012) Snakemake: a scalable bioinformatics workflow engine. Bioinformatics 28(19):2520–2522
20. Larsson A (2014) AliView: a fast and lightweight alignment viewer and editor for large datasets. Bioinformatics 30(22):3276–3278

Chapter 11

Analysis of Protein Intermolecular Interactions with MAFFT-DASH

John Rozewicki, Songling Li, Kazutaka Katoh, and Daron M. Standley

Abstract

The Database of Aligned Structural Homologs (DASH) is a tool for efficiently navigating the Protein Data Bank (PDB) by means of pre-computed pairwise structural alignments. We recently showed that, by integrating DASH structural alignments with the multiple sequence alignment (MSA) software MAFFT, we were able to significantly improve MSA accuracy without dramatically increasing manual or computational complexity. In the latest DASH update, such queries are not limited to PDB entries but can also be launched from user-provided protein coordinates. Here, we describe a further extension of DASH that retrieves intermolecular interactions of all structurally similar domains in the PDB to a query domain of interest. We illustrate these new features using a model of the NYN domain of the ribonuclease N4BP1 as an example. We show that the protein-nucleotide interactions returned are distributed on the surface of the NYN domain in an asymmetric manner, roughly centered on the known nuclease active site.

Key words Protein-nucleotide interaction, Protein-protein interaction, Protein structural alignment, RNA structure, Ribonuclease, Binding site prediction, Database query, Structural domain

1 Introduction

It is well-known that protein domains sharing a similar overall fold can exhibit very low similarity in their primary sequences. Less is known about the conservation of intermolecular binding sites. Because experimentally determined protein structures often include other bound molecules, the conservation of intermolecular interactions can be observed through use of structural alignments. Since more and more structural assemblies are being solved recently (e.g., by cryo-EM), it is expected that the number of "structurally conserved interactions" that can be extracted from the Protein Data Bank (PDB) will increase in the near future. Here, we propose a method for automatically identifying such interactions and mapping them back on a protein of interest.

There are potentially many steps, involving multiple parameters and decisions, that are required when searching for structurally conserved interactions: proteins must be decomposed into domains, domains must be aligned, intermolecular interfaces must be identified, etc. DASH itself is a rather simple tool: given a query structural domain, DASH returns a ranked list of structurally similar domains in the PDB along with their alignments and structural superpositions to the query. Thus, in order to automate the retrieval of structurally conserved interactions, DASH features have been expanded, and new workflow scripts have been developed in an effort to simplify these complex tasks.

In DASH, the computationally expensive structural alignments have been pre-computed for every representative domain in the PDB, allowing chain-level and domain-level alignments to be accessed with minimal overhead. The MAFFT-DASH server converts structurally conserved residues into constraints for MSA calculation [1]. The positive effect of structural information on MSA accuracy has been conclusively demonstrated [2–4]. Here, we use this approach to retrieve both structural alignments and intermolecular interactions, which we map back onto the query domain. This is accomplished, in part, by leveraging two new DASH features: (i) the option to Search by Structure, which allows non-PDB entries to be used as input, and (ii) export of the transformation matrix, which allows any molecule in the reference frame of a structural hit to be rotated into the reference frame of the query. To illustrate the use of these methods on the command line, we provide a Python script, **`get_interactions.py`**, `which` accepts a user-provided domain structure, queries DASH to find structural hits and their transformation matrices, and then applies the transformations to all molecules interacting with the hits. We prepared a second script, **`calculate_conservation.py`**, `which computes conservation of the query domain and` formats the output for visualization. We illustrate the use of these new tools by analyzing a model of the NYN domain-containing protein, Nedd4 binding protein 1 (N4BP1). Supplemental materials for this chapter are available at https://sysimm.org/dash/mimb2020. This includes Python scripts, example inputs/outputs, and system requirements.

2 NYN Domain-Containing RNases

A practical example of how structural similarity can be used for functional inference is the NYN domain. This domain was originally identified by phylogenetic analysis of several mammalian proteins [5]. One of these proteins, ZC3H12A (also known as

MCPIP-1), was originally proposed to act as a transcription factor [6]. However, by combining experimental and structural modeling data, ZC3H12A was subsequently shown to be a ribonuclease acting preferentially on a set of pro-inflammatory cytokine messenger RNAs [7]. We note that the structural model in this case was built on a template in the "twilight zone" (13% sequence identity) which itself was a structural genomics target with no known function (PDB ID 2QIP), highlighting the importance of structural alignment in remote-homology function prediction. ZC3H12A has since been shown to function in a variety of cell types as an essential posttranscriptional regulator and has thus been renamed "Regnase-1" [8]. Recently, suppression of Regnase-1 expression in T cells was found to be critical for controlling tumor growth [9].

Despite these findings, very little is known about how Regnase-1 recognizes its target RNA. A crystal structure of nucleotide-free Regnase-1 has been solved [10], and residues essential for RNase activity and RNA binding have been identified [11]. However, there is yet to be a report of Regnase-1 bound to RNA. Since there are many other proteins which contain NYN-like domains in the PDB, some of which contain bound nucleotides, we hypothesized that a general picture of how NYN domains interact with nucleotides could be realized through a combination of structural alignment and mapping of structurally conserved interactions.

A second NYN-containing RNase is N4BP1, which was originally reported to interact with the E3 ubiquitin ligase Nedd4 [12]. The presence of an NYN domain suggested that N4BP1 might also play a role in regulating RNA levels. Indeed, it was recently shown that the catalytic residues are conserved between N4BP1 and Regnase-1; furthermore, N4BP1 was demonstrated to degrade HIV RNA in CD4+ T cells [13]. To date, an experimentally determined structure of N4BP1 has not been reported. To illustrate the use of DASH in visualizing putative protein-nucleotide interactions, we work through the steps needed to build a 3D model of the N4BP1 NYN domain, collect the top structural homologs to this model, and examine structurally conserved protein-protein and protein-nucleotide interactions.

3 Construction of N4BP1 NYN Domain Homology Model

We prepared a 3D model of the N4BP1 NYN domain (residues 615–775, red font) based on the Regnase-1 NYN domain (PDB ID 3V32).

```
MAARAVLDEFTAPAEKAELLEQSRGRIEGLFGVSLAVLGALGAEEPLPARIWLQLCGAQEAVH

SAKEYIKGICEPELEERECYPKDMHCIFVGAESLFLKSLIQDTCADLCILDIGLLGIRGSAEA

VVMARSHIQQFVKLFENKENLPSSQKESEVKREFKQFVEAHADNYTMDLLILPTSLKKELLTL

TQGEENLFETGDDEVIEMRDSQQTEFTQNAATGLNISRDETVLQEEARNKAGTPVSELTKQMD

TVLSSSPDVLFDPINGLTPDEEALSNERICQKRRFSDSEERHTKKQFSLENVQEGEILHDAKT

LAGNVIADLSDSSADSENLSPDIKETTEEMEYNILVNFFKTMGYSQEIVEKVIKVYGPSTEPL

LLLEEIEKENKRFQEDREFSAGTVYPETNKTKNKGVYSSTNELTTDSTPKKTQAHTQQNMVEK

FSQLPFKVEAKPCTSNCRINTFRTVPIEQKHEVWGSNQNYICNTDPETDGLSPSVASPSPKEV

NFVSRGASSHQPRVPLFPENGLHQQPEPLLPNNMKSACEKRLGCCSSPHSKPNCSTLSPPMPL

PQLLPSVTDARSAGPSDHIDSSVTGVQRFRDTLKIPYKLELKNEPGRTDLKHIVIDGSNVAIT

HGLKKFFSCRGIAIAVEYFWKLGNRNITVFVPQWRTRRDPNVTEQHFLTQLQELGILSLTPAR

MVFGERIASHDDRFLLHLADKTGGIIVTNDNFREFVNESVSWREIITKRLLQYTFVGDIFMVP

DDPLGRSGPRLEEFLQKEVCLRDMQPLLSALPNVGMFDPSFRVPGTQAASTSHQPPTRIQGAP

SSHWLPQQPHFPLLPALPSLQ

QNLPMPAQRSSAETNELREALLKIFPDSEQRLKIDQILVAHPYMKDLNALSAMVLD
```

We then used the Spanner [14] server (https://sysimm.org/spanner/) to build a 3D model based on a pairwise sequence alignment to a structural template (*see* Fig. 1).

4 Preparation of a Ranked List of NYN-Like Domains Using DASH

We next used the new Search by Structure feature in DASH that allows user-provided structures to be efficiently aligned to DASH domains. On the web interface, Search by Structure can be invoked by uploading a PDB or mmCIF file. The input and results page are illustrated using the model of the N4BP1 NYN domain (*see* Fig. 2). All of the output can be downloaded as an easily parsable zip archive of JSON data. The superimposed structures can be downloaded as a 7zip archive. A table giving an overview of the results can also be downloaded as a TSV file.

A

```
N4BP1      TDLKHIVIDGSNVAITHGLKKFFSCRGIAIAVEYFWKLGNRNITVFVPQWRT---RRDPNVTEQHFLTQLQELGILSLTPARMVFGE
Regnase-1  SDLRPVVIDGSNVAMSHGNKEVFSCRGILLAVNWFLERGHTDITVFVPSWRKEQPRPDVPITDQHILRELEKKKILVFTPSRRVGGK

N4BP1      RIASHDDRFLLHLADKTGGIIVTNDNFREFVNESVSWREIITKRLLQYTFVGDIFMVPDDPLGRSGPRLEEFLQKE
Regnase-1  RVVCYDDRFIVKLAYESDGIVVSNDTYRDLQGERQEWKRFIEERLLMYSFVNDKFMPPDDPLGRHGPSLDNFLRKK
```

B

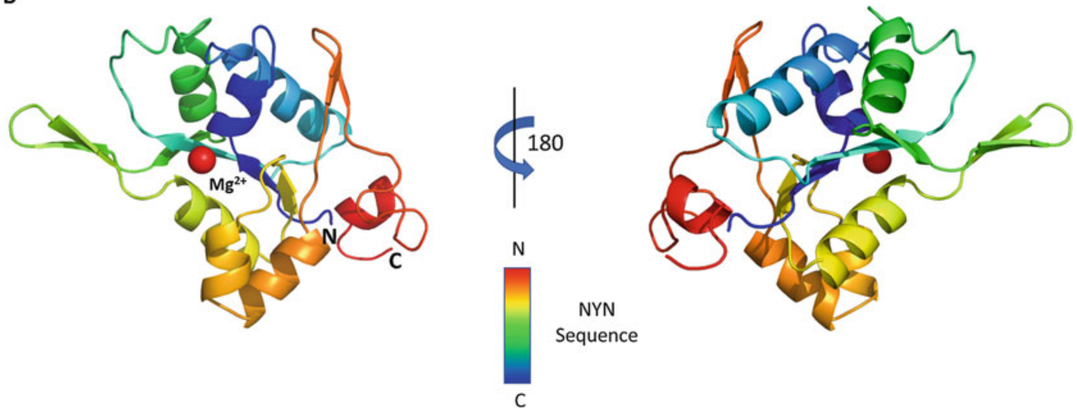
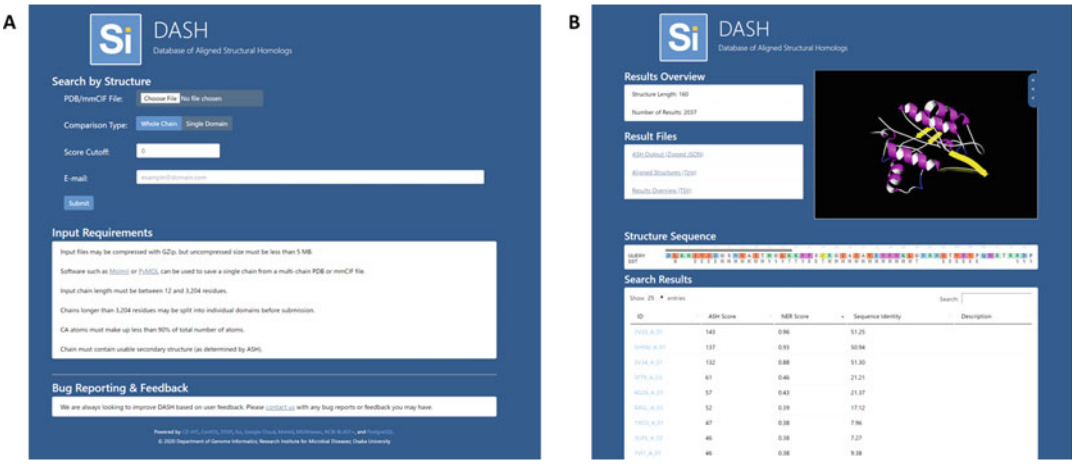


Fig. 1 N4BP1 model. A, Sequence alignment between N4BP1 and Regnase-1 NYN domains; B, 3D model of N4BP1 NYN domain with sphere showing approximate location of catalytic Mg^{2+} in red



Search by Structure

PDB/mmCIF File: No file chosen

Comparison Type: Whole Chain

Score Cutoff: 0

E-mail: example@domain.com

Input Requirements

Input files may be compressed with GZip, but uncompressed size must be less than 5 MB.

Software such as Molmil or PyMOL can be used to save a single chain from a multi-chain PDB or mmCIF file.

Input chain length must be between 12 and 3,204 residues.

Chains longer than 3,204 residues may be split into individual domains before submission.

CA atoms must make up less than 90% of total number of atoms.

Chain must contain usable secondary structure (as determined by ASH).

Bug Reporting & Feedback

We are always looking to improve DASH based on user feedback. Please contact us with any bug reports or feedback you may have.

Powered by [illegible]

© 2020 Department of Genome Informatics, Research Institute for Microbial Diseases, Osaka University

Results Overview

Structure Length: 160

Number of Results: 2037

Result Files

ASH Output (Zipped JSON)

Aligned Structures (Tgz)

Results Overview (TSV)

Structure Sequence

Search Results

ID	ASH Score	NER Score	Sequence Identity	Description
[illegible]	143	0.96	51.25	
[illegible]	137	0.93	50.94	
[illegible]	132	0.88	51.30	
[illegible]	61	0.46	21.21	
[illegible]	57	0.43	21.37	
[illegible]	52	0.39	17.12	
[illegible]	47	0.38	7.96	
[illegible]	46	0.38	7.27	
[illegible]	46	0.38	9.38	

Fig. 2 DASH web server. The input (**a**) and results (**b**) pages on the DASH web server

5 Extracting Putative Nucleotide Interactions from DASH Hits

In order to collect the intermolecular and interdomain interactions of the DASH hits and map them onto the query domain, we must carry out a number of steps as follows:

1. Read the DASH hits in descending order.

2. Extract the PDB ID and domain ID of the hit, along with the transformation matrix needed to superimpose the hit-domain onto the query-domain.
3. Download the associated mmCIF file from the PDB. We use mmCIF files for this step because they contain cleaner and more complete data and can be parsed easily by standard tools, such as BioPython (https://biopython.org/) [15].
4. Extract "environment" coordinates: interactions between the hit-domain and other domains, nucleotides, heteroatoms, etc. from the entire PDB entry.
5. Superimpose the environment onto the query-domain using the transformation matrix.
6. Filter and output the superimposed environment according to the user's specifications.

Because the steps above would be tedious to perform manually, we have prepared a Python script (**`get_interactions.py`**) that automates them. In its simplest form, the script takes a DASH domain ID (or Search by Structure job ID) as input:

```
get_interactions.py -qd dash503929320
```

Job IDs currently expire after 2 weeks. The current set of options for this script are as follows:

```
usage: get_interactions.py [-h] [-rest REST] [-verbose] -qd
QUERY_ID
                             [-td TEMPLATE_ID] [-cores CORES] [-limit
LIMIT]
                                    [-filter-score FILTER_SCORE] [-o
OUTFILE] [-prot]
                                 [-nuc] [-het] [-water] [-prot-close
PROT_CLOSE]
                                [-prot-clash PROT_CLASH] [-het-close
HET_CLOSE]
                                  [-het-clash HET_CLASH] [-nuc-close
NUC_CLOSE]
                                [-nuc-clash NUC_CLASH] [-water-close
WATER_CLOSE]
                               [-water-clash WATER_CLASH] [-ftp FTP]
                                  [-cache CACHE] [-display-template-
chain]

optional arguments:
  -h, --help            Show this help message and exit
  -rest REST             Base URL to use for querying the DASH
```

```
REST interface.
  -verbose              Output many more status messages.
   -qd QUERY_ID            Query DASH domain ID or Search by
Structure job ID.
  -td TEMPLATE_ID        Template DASH domain ID. Ignored when
-qd is a Search
                        by Structure job ID.
  -cores CORES          Number of CPU cores to use.
  -limit LIMIT         At most, this many superimposed entries
will be
                        output. 0 means no limit.
  -filter-score FILTER_SCORE
                      Alignments with scores below this cutoff
will be not
                        be used.
  -o OUTFILE            Output mmCIF file.
  -prot                 Show protein residues in output.
  -nuc                  Show nucleotides in output.
  -het                  Show HETATM in output.
  -water                Show water HETATM in output.
  -prot-close PROT_CLOSE
                    Protein atoms less than this cutoff will be
considered
                        close.
  -prot-clash PROT_CLASH
                    Protein atoms less than this cutoff will be
considered
                        clashing.
   -het-close HET_CLOSE  HETATM less than this cutoff will be
considered close.
   -het-clash HET_CLASH  HETATM less than this cutoff will be
considered
                        clashing.
  -nuc-close NUC_CLOSE  Nucleotides less than this cutoff will
be considered
                        close.
  -nuc-clash NUC_CLASH  Nucleotides less than this cutoff will
be considered
                        clashing.
  -water-close WATER_CLOSE
                          Water less than this cutoff will be
considered close.
  -water-clash WATER_CLASH
                          Water less than this cutoff will be
considered
                        clashing.
  -ftp FTP               PDB FTP mirror to use for downloading
mmCIF files.
```

```
  -cache CACHE         Temporary folder to use for mmCIF file
cache.
  -display-template-chain
                       Display template chain from alignment.
```

DASH hits include not only alignments and top-line score information but also the transformation matrix that can be used to superimpose the hits onto the query structure. This means that not only the specific protein residues in the alignment, but coordinates for other chains or molecules in the PDB entry can also be overlaid on the query. This is particularly useful for visualization of the potential interactions between N4BP1 and nucleotides or other protein chains/domains.

In order to search exclusively for protein-nucleotide interactions for the NYN domain in N4BP1, we executed the following command:

```
get_interactions.py -qd dash503929320 -filter-score 20 -nuc -o
out-nuc.cif.gz
```

We then visualized the output in PyMOL. Inside PyMOL, nucleotide chains were selected by the following command in the console window:

```
select nucleic, (byres polymer & name P)
```

Then, using the pull-down menu, these nucleotide chains were displayed as orange ribbons with gray mesh, while the query protein was displayed as a cartoon. The Mg^{2+} from the Regnase-1 crystal structure (PDB ID 3 V33) was also incorporated as a red sphere. The result shows very dramatically that the distribution of nucleotides surrounding the NYN domains is highly asymmetric (*see* Fig. 3). Moreover, we can see the active site Mg^{2+} appears roughly in the center of the cloud of nucleotides.

It is worth looking at the biochemical function of the DASH hits (*see* Table 1). They can be divided roughly evenly into RNA- and DNA-binding proteins. Most are annotated as nucleases, but two are annotated as ribosomal GTPases, one as a glutaminyl-tRNA synthetase and one as an isopentenyl-tRNA transferase. Of particular interest is the endoribonuclease UTP24, which cleaves pre-rRNA at a conserved pseudoknot [16]. Interestingly, in a recently solved cryo-EM structure, the cleavage site is located far from the active site of UTP24, suggesting that a yet-unknown

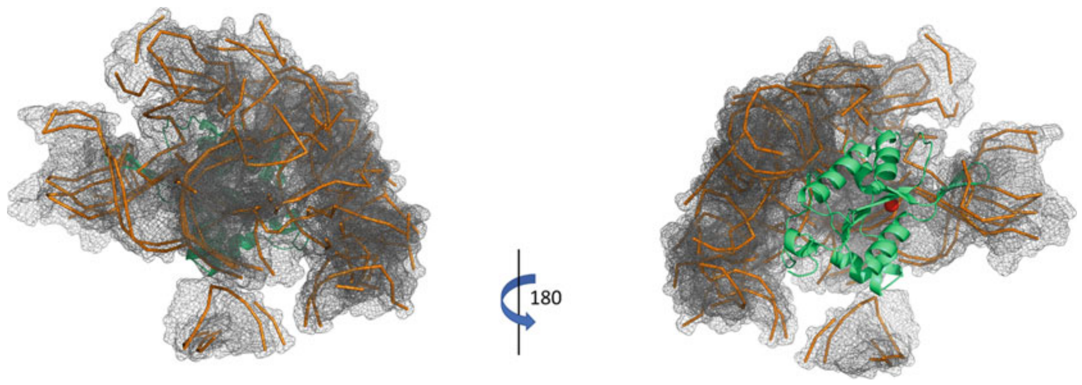


Fig. 3 Distribution of nucleotides. The distribution of nucleotides around the N4BP1 NYN domains is highly asymmetric

Table 1
NYN-nucleotide hits

Score	PDB ID	Nucleotide	Function
38	3ZDB_A	DNA	Exonuclease IX
37	5OQL_P	RNA	Endonuclease UTP24
32	5WLC_SL	RNA	Endonuclease UTP24
27	5T9J_A	DNA	Endonuclease GEN1
27	5CO8_A	DNA	Endonuclease GEN1
25	6GRC_A	DNA	Endonuclease GEN1
23	4A2I_V	RNA	Ribosomal GTPase RSGA
21	5V07_Z	DNA	Exonuclease Exo1
21	1QTQ_A	RNA	tRNA synthetase
20	3FOZ_A	RNA	tRNA transferase
20	2YKR_W	RNA	Ribosomal GTPase RSGA

helicase may be required for unwinding or remodeling the RNA secondary structure [17]. This result is reminiscent of the helicase UPF1, which "licenses" Regnase-1 by unwinding target stem-loop structures [18].

6 Visualizing NYN-Nucleotide Interactions Along with Sequence Conservation or RNA-Binding Propensity

A MAFFT alignment of N4BP1 homologs can be computed by a combination of BLAST and MAFFT. First, we selected the top 1000 nonredundant sequence homologs to the N4BP1 NYN domain using BLAST+. Then, we computed an MSA of these sequences using MAFFT. The residue conservation was next calculated as a percentage of the BLOSUM62 substitution matrix for each residue in the MSA relative to the maximum BLOSUM62 score of each column. This percentage was then inverted and written into the b-factor column of the query protein mmCIF file. These steps have been automated as a Python script, **`calculate_conservation.py`**. The usage of the script is as follows:

```
usage: calculate_conservation.py [-h] [-verbose] -i PDB_PATH
                                    [-temp TEMP_FOLDER] [-top
BLAST_TOP_N]
                                 [-o OUT_PATH]

optional arguments:
  -h, --help          show this help message and exit
  -verbose            Output many more status messages.
  -i PDB_PATH         Path to PDB file.
  -temp TEMP_FOLDER   Path to folder to use for temporary data.
   -top BLAST_TOP_N    Use at most the top n hits from BLAST
search of nr
                      database.
  -o OUT_PATH         Output mmCIF file
```

In our N4BP1 example, we used the following command:

```
calculate_conservation.py -i n4bp1-spanner-model.pdb -top 1000
-o n4bp1_conc.cif.gz
```

The resulting file, **`n4bp1_conc.cif.gz`,** when viewed in PyMOL, shows that there is no obvious correlation between residue conservation and nucleotide contact (*see* Fig. 4). However, when we instead predict RNA-binding sites using the aaRNA [19] web server (https://sysimm.org/aarna/), we found that the nucleotide-contacting residues had much higher RNA-binding propensities than the non-contacting regions (*see* Fig. 5).

Taken together, the nucleotide density and RNA-binding propensity suggest that the DASH hits are not arbitrary, but appear to cluster around RNA-binding residues near the active site. On the other hand, we can observe many conserved residues that appear not to make contact with nucleotides. One possibility here would

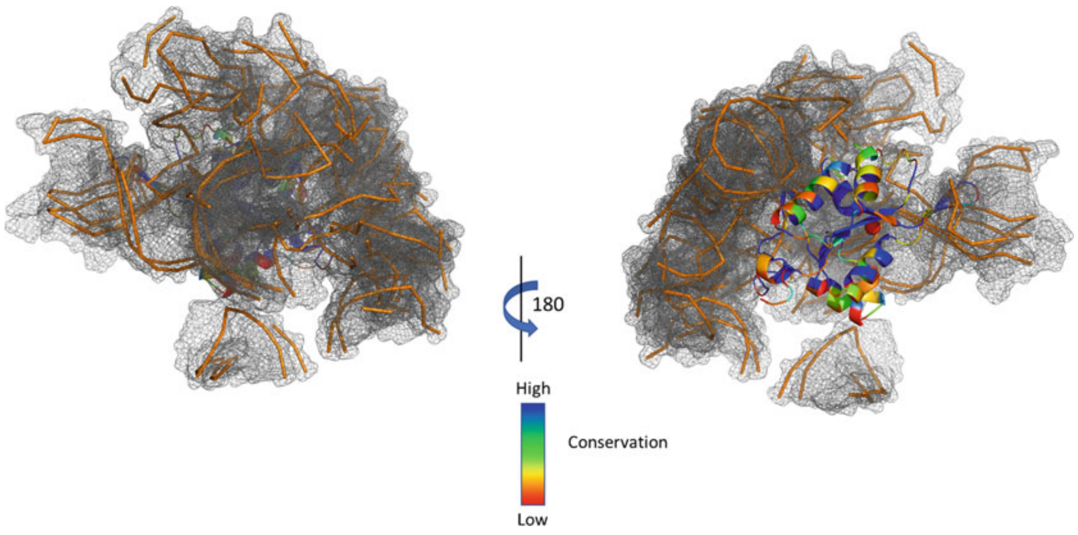


Fig. 4 N4BP1 NYN domain residue conservation. Residues near the active site and hydrophobic core are conserved, as expected. However, the residue conservation did not appear to correlate with the density of nucleotides around the N4BP1 NYN domain

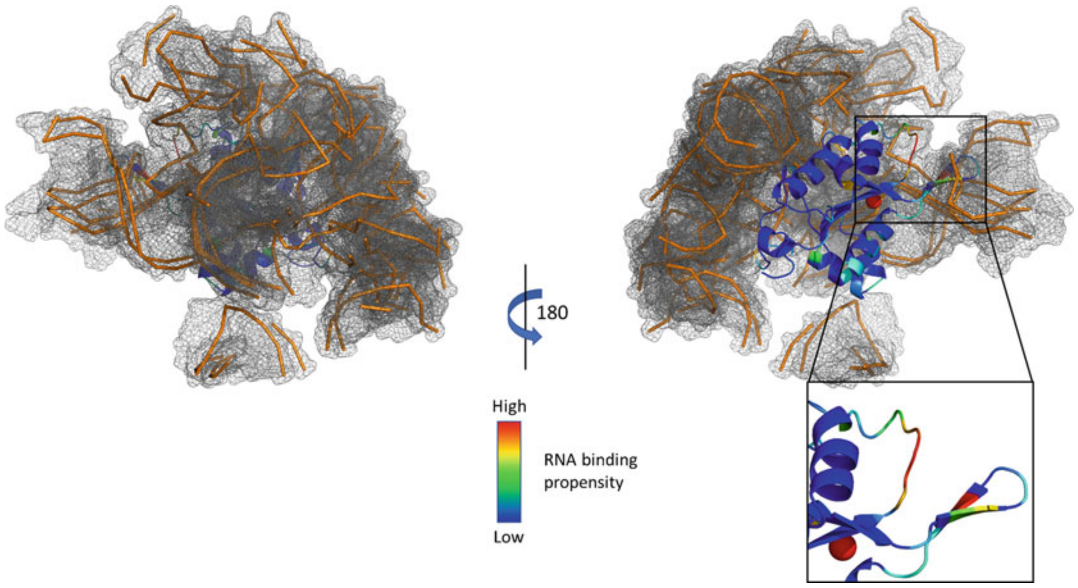


Fig. 5 RNA-binding propensity of N4BP1. Residues in the N4BP1 NYN domain that are predicted to have high RNA-binding propensity appear buried in high nucleotide density regions

be that such conserved surface residues are involved in protein-protein interactions. To investigate this idea further, we next assessed the density of proteins in the environments of other NYN domain-containing proteins (*see* Table 2).

Table 2
NYN-interacting proteins

Score	PDB ID	Protein chains	Protein names
37	5OQL_P	C,R,T,Z,a,d,r,w,0	Utp3 (something about silencing protein 10), Nop1 (rRNA methyltransferase), nucleolar protein 56, S28-like, Imp3, Sof1, S9-like,S22-like,Faf1
32	5WLC_SL	L9,LE,LU,NB,NC, NE, SA, SC, SF, SI	S9, S22, Sof1, Sas10, Lcp5, Faf1,Nop56,Nop1,Snu13, Bms1
23	4A2I_V	L	S12
20	2YKR_W	K,L	S11, S12

7 Extracting Putative Protein Interactions from DASH Hits

The extraction of protein-protein interactions is analogous to that of protein-nucleotide interactions and can be accomplished with the same script by using the **-prot** command-line option. In the case of N4BP1, simply giving the **-prot** option will lead to a large number of unrelated hits due to the fact that NYN-like domains occur in a wide range of proteins unrelated to nucleotide binding (such as protein synthesis or hydrolysis). If we want to see only the protein interactions for those proteins also involved in interactions with nucleotides, we can give both the **-nuc** and the **-prot** options.

The command we used is as follows:

```
get_interactions.py -qd dash503929320 -filter-score 20 -nuc
-prot -o out-prot.cif.gz
```

The resulting hits are shown in Fig. 6. As we can see, the environment surrounding the NYN domains not populated with nucleotides is highly populated with proteins. These results suggest that NYN domains involved in nucleotide interactions often function as part of larger complexes.

A quick look at the domains obtained in this exercise suggests that they are all proteins that directly interact with ribosomal RNA. In Fig. 7, the individual ribosomal protein complexes associated with nucleotide-binding NYN domains are shown. The proteins appear to be rather evenly distributed around the NYN domain. Moreover, these are close contacts between the surrounding proteins and RNA. Given these observations, the asymmetry of the nucleotides that interact with the NYN domain is likely due to the spatial requirements for NYN nuclease activity.

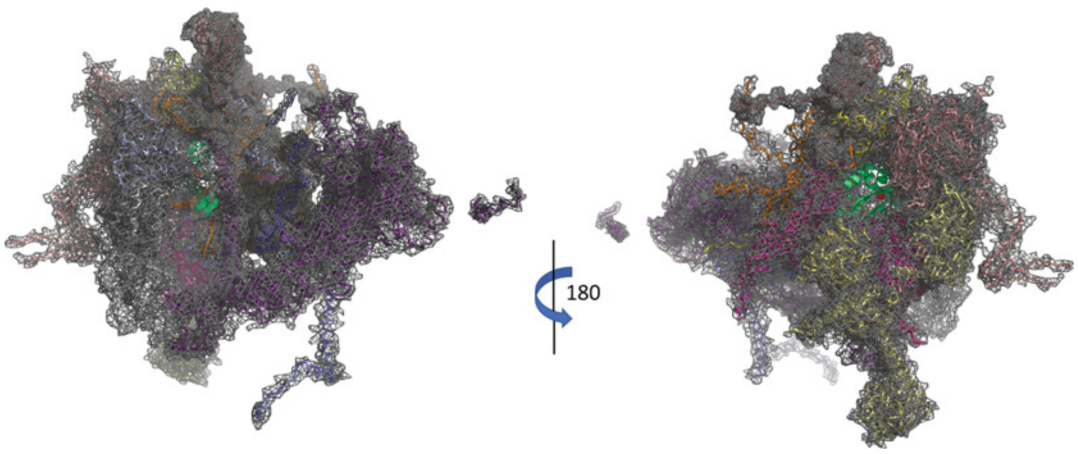


Fig. 6 Distribution of proteins. The distribution of proteins contacting NYN domains with bound nucleotides is shown. In contrast to the case of nucleotides, the NYN domain is completely obscured by protein-protein interactions

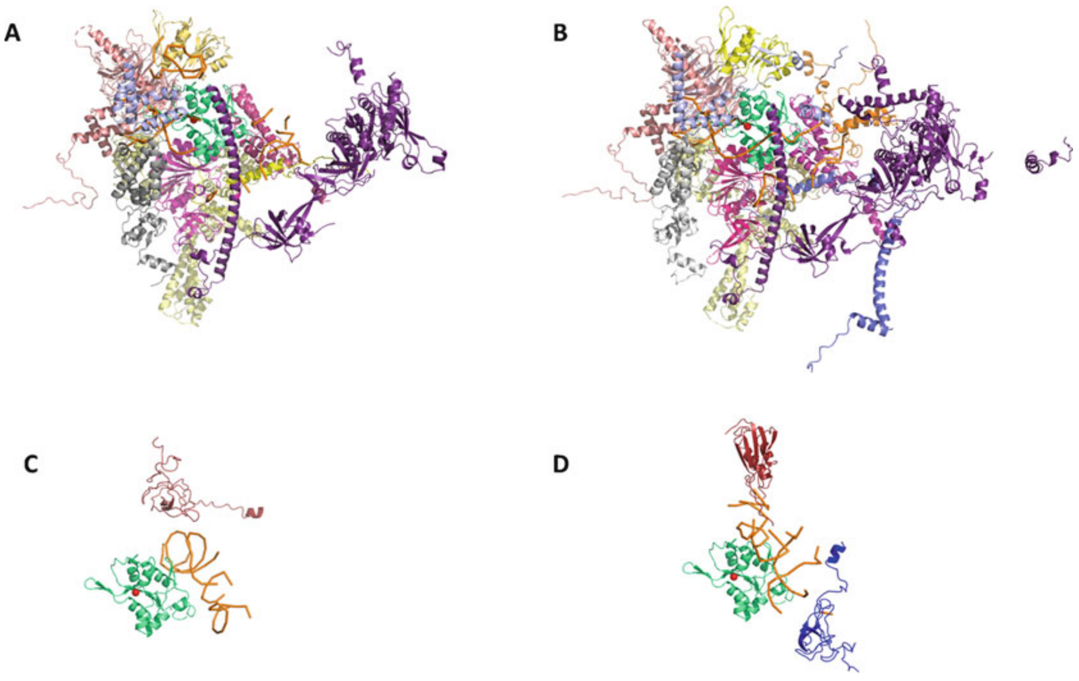


Fig. 7 Ribosomal proteins bound to top NYN DASH hits. The top four DASH hits are shown as cartoons. In **a**–**d** the NYN domain is shown in the orientation that matches that of the left side of Figs. 3–6, while in **e**–**h**, the orientation matches that of the right-hand side of Figs. 3–6

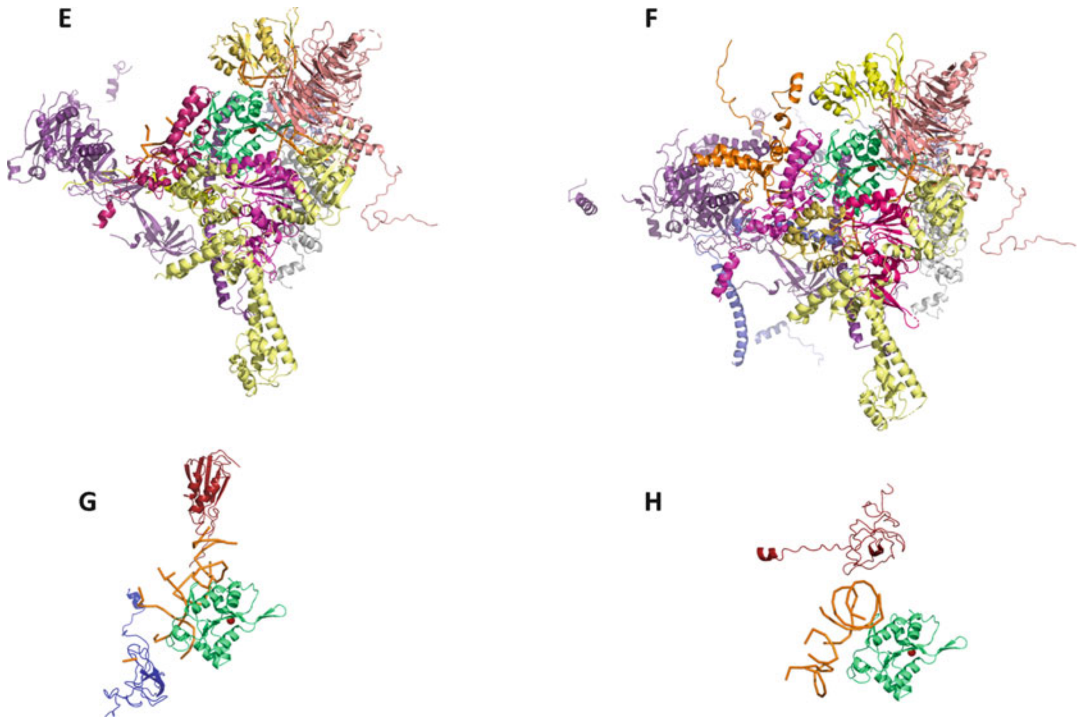


Fig. 7 (continued)

8 Conclusions

The tools described in this chapter illustrate the use of DASH and MAFFT for analyzing structurally conserved interactions. Changes in the parameters may be needed for specific problems. In particular, the optimal `-filter-score` will depend on the domain of interest. The ability of DASH to quickly search all representative PDB entries will be particularly powerful as the number of intermolecular assemblies (i.e., determined by cryo-EM) increases.

References

1. Rozewicki J, Li S, Amada KM, Standley DM, Katoh K (2019) MAFFT-DASH: integrated protein sequence and structural alignment. Nucleic Acids Res 47(W1):W5–W10. https://doi.org/10.1093/nar/gkz342
2. Armougom F, Moretti S, Poirot O, Audic S, Dumas P, Schaeli B, Keduas V, Notredame C (2006) Expresso: automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee. Nucleic Acids Res 34(Web Server issue):W604–W608. https://doi.org/10.1093/nar/gkl092
3. Di Tommaso P, Moretti S, Xenarios I, Orobitg M, Montanyola A, Chang JM, Taly JF, Notredame C (2011) T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension. Nucleic Acids Res 39 (Web Server issue):W13–17. https://doi.org/10.1093/nar/gkr245
4. Pei J, Kim BH, Grishin NV (2008) PROMALS3D: a tool for multiple protein sequence and structure alignments. Nucleic Acids Res 36 (7):2295–2300. https://doi.org/10.1093/nar/gkn072
5. Anantharaman V, Aravind L (2006) The NYN domains: novel predicted RNAses with a PIN

domain-like fold. RNA Biol 3(1):18–27. https://doi.org/10.4161/rna.3.1.2548

6. Niu J, Azfer A, Zhelyabovska O, Fatma S, Kolattukudy PE (2008) Monocyte chemotactic protein (MCP)-1 promotes angiogenesis via a novel transcription factor, MCP-1-induced protein (MCPIP). J Biol Chem 283(21):14542–14551. https://doi.org/10.1074/jbc.M802139200
7. Matsushita K, Takeuchi O, Standley DM, Kumagai Y, Kawagoe T, Miyake T, Satoh T, Kato H, Tsujimura T, Nakamura H, Akira S (2009) Zc3h12a is an RNase essential for controlling immune responses by regulating mRNA decay. Nature 458(7242):1185–1190. https://doi.org/10.1038/nature07924
8. Iwasaki H, Takeuchi O, Teraguchi S, Matsushita K, Uehata T, Kuniyoshi K, Satoh T, Saitoh T, Matsushita M, Standley DM, Akira S (2011) The IkappaB kinase complex regulates the stability of cytokine-encoding mRNA induced by TLR-IL-1R by controlling degradation of regnase-1. Nat Immunol 12(12):1167–1175. https://doi.org/10.1038/ni.2137
9. Wei J, Long L, Zheng W, Dhungana Y, Lim SA, Guy C, Wang Y, Wang YD, Qian C, Xu B, Kc A, Saravia J, Huang H, Yu J, Doench JG, Geiger TL, Chi H (2019) Targeting REGNASE-1 programs long-lived effector T cells for cancer therapy. Nature 576(7787):471–476. https://doi.org/10.1038/s41586-019-1821-z
10. Xu J, Peng W, Sun Y, Wang X, Xu Y, Li X, Gao G, Rao Z (2012) Structural study of MCPIP1 N-terminal conserved domain reveals a PIN-like RNase. Nucleic Acids Res 40(14):6957–6965. https://doi.org/10.1093/nar/gks359
11. Yokogawa M, Tsushima T, Noda NN, Kumeta H, Enokizono Y, Yamashita K, Standley DM, Takeuchi O, Akira S, Inagaki F (2016) Structural basis for the regulation of enzymatic activity of Regnase-1 by domain-domain interactions. Sci Rep 6:22324. https://doi.org/10.1038/srep22324
12. Murillas R, Simms KS, Hatakeyama S, Weissman AM, Kuehn MR (2002) Identification of developmentally expressed proteins that functionally interact with Nedd4 ubiquitin ligase. J Biol Chem 277(4):2897–2907. https://doi.org/10.1074/jbc.M110047200
13. Yamasoba D, Sato K, Ichinose T, Imamura T, Koepke L, Joas S, Reith E, Hotter D, Misawa N, Akaki K, Uehata T, Mino T, Miyamoto S, Noda T, Yamashita A, Standley DM, Kirchhoff F, Sauter D, Koyanagi Y, Takeuchi O (2019) N4BP1 restricts HIV-1 and its inactivation by MALT1 promotes viral reactivation. Nat Microbiol 4(9):1532–1544. https://doi.org/10.1038/s41564-019-0460-3
14. Lis M, Kim T, Sarmiento J, Kuroda D, Dinh H, Kinjo AR, Devadas S, Nakamura H, Standley DM (2011) Bridging the gap between single-template and fragment based protein structure modeling using Spanner. Immunome Research 7(1)
15. Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, de Hoon MJ (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics 25(11):1422–1423. https://doi.org/10.1093/bioinformatics/btp163
16. Wells GR, Weichmann F, Colvin D, Sloan KE, Kudla G, Tollervey D, Watkins NJ, Schneider C (2016) The PIN domain endonuclease Utp24 cleaves pre-ribosomal RNA at two coupled sites in yeast and humans. Nucleic Acids Res 44(11):5399–5409. https://doi.org/10.1093/nar/gkw213
17. Cheng J, Kellner N, Berninghausen O, Hurt E, Beckmann R (2017) 3.2-A-resolution structure of the 90S preribosome before A1 pre-rRNA cleavage. Nat Struct Mol Biol 24(11):954–964. https://doi.org/10.1038/nsmb.3476
18. Mino T, Iwai N, Endo M, Inoue K, Akaki K, Hia F, Uehata T, Emura T, Hidaka K, Suzuki Y, Standley DM, Okada-Hatakeyama M, Ohno S, Sugiyama H, Yamashita A, Takeuchi O (2019) Translation-dependent unwinding of stem-loops by UPF1 licenses Regnase-1 to degrade inflammatory mRNAs. Nucleic Acids Res 47(16):8838–8859. https://doi.org/10.1093/nar/gkz628
19. Li S, Yamashita K, Amada KM, Standley DM (2014) Quantifying sequence and structural features of protein-RNA interactions. Nucleic Acids Res 42(15):10086–10,098. https://doi.org/10.1093/nar/gku681

Mustguseal and Sister Web-Methods: A Practical Guide to Bioinformatic Analysis of Protein Superfamilies

Dmitry Suplatov, Yana Sharapova, and Vytas Švedas

Abstract

Bioinformatic analysis of functionally diverse superfamilies can help to study the structure-function relationship in proteins, but represents a methodological challenge. The Mustguseal web-server can build large structure-guided sequence alignments of thousands of homologs that cover all currently available sequence variants within a common structural fold. The input to the method is a PDB code of the query protein, which represents the protein superfamily of interest. The collection and subsequent alignment of protein sequences and structures is fully automated and driven by the particular choice of parameters. Four integrated sister web-methods—the Zebra, pocketZebra, visualCMAT, and Yosshi—are available to further analyze the resulting superimposition and identify conserved, subfamily-specific, and co-evolving residues, as well as to classify and study disulfide bonds in protein superfamilies. The integration of these web-based bioinformatic tools provides an out-of-the-box easy-to-use solution, first of its kind, to study protein function and regulation and design improved enzyme variants for practical applications and selective ligands to modulate their functional properties. In this chapter, we provide a step-by-step protocol for a comprehensive bioinformatic analysis of a protein superfamily using a web-browser as the main tool and notes on selecting the appropriate values for the key algorithm parameters depending on your research objective. The web-servers are freely available to all users at https://biokinet.belozersky.msu.ru/m-platform with no login requirement.

Key words Bioinformatic analysis, Protein superfamilies, Multiple alignment, Conserved positions, Specific positions, Co-evolving positions, Disulfide bonds

1 Introduction

Understanding the relationship between protein sequence/structure and its biological function is one of the most complex problems in modern biology. During evolution of proteins from a common ancestor, one functional property may be preserved, while others may vary as a result of mutations introduced into the protein structure, leading to functional diversity. Comparative analysis of homologs implementing different properties within a common structure of the superfamily can help to understand the relationship between the protein structure, function, and regulation [1], but represent a methodological and computational challenge, as both sequences and structures have to be taken into account to accurately superimpose evolutionarily distantly related proteins [2–4]. Only a handful of tools are available to address this issue, MAFFT-DASH [2] and Mustguseal [3] being the most recent ones. The MAFFT-DASH alignment tool supports a situation where there is no prior information about 3D-structures of homologs. Mustguseal is useful when the analysis of a specific query protein in the context of the corresponding superfamily is of interest (e.g., for the purpose of protein engineering or annotation of novel drug-binding sites).

Mustguseal and integrated sister web-servers feature a collection of open-access methods available at https://biokinet.belozersky.msu.ru/m-platform. The aim of this online platform is to provide an easy-to-use comprehensive solution for the systematic bioinformatic analysis of protein superfamilies. The key web-server Mustguseal can automatically collect and align thousands of sequences and structures of proteins within a superfamily to produce a large structure-guided sequence alignment. Four types of bioinformatic algorithms—i.e., for the database search and multiple alignment by sequence and 3D-structure comparison—are implemented to take into account the vast variability of proteins within a superfamily: superimposition of the protein 3D-structures, known to be more conserved among homologs throughout the evolution, is used to match distant relatives, whereas alignment of amino acid sequences is carried out to match close homologs. The Mustguseal protocol is initiated by a query protein 3D-structure (i.e., a member of the corresponding superfamily) and consists of four major steps (Fig. 1). First, the structure similarity search by the SSM algorithm [5] is implemented to collect evolutionarily distantly related proteins that lost sequence similarity during natural selection and specialization from a common ancestor. These representative proteins are expected to introduce different protein families into the alignment. Then, a 3D-superimposition of the collected structures is performed by the MATT algorithm [4, 6] to create the core 3D-structural alignment. Each representative protein is used as a query to run a sequence similarity search by the BLAST and collect evolutionarily close relatives—members of the corresponding families. Protein sequences within these collections are further aligned using the MAFFT [7]. Finally, the structure-guided superimposition of all the collected sequences is performed using the core 3D-structural alignment as a guide. The characteristics of the alignment—its diversity and scope, redundancy (i.e., presence of similar sequences), size/thickness (number of proteins), length (number of columns), etc.—are driven by various parameters selected by the user when running the Mustguseal algorithm. Generally speaking, no particular choice of thresholds for these parameters can be recommended in advance as they

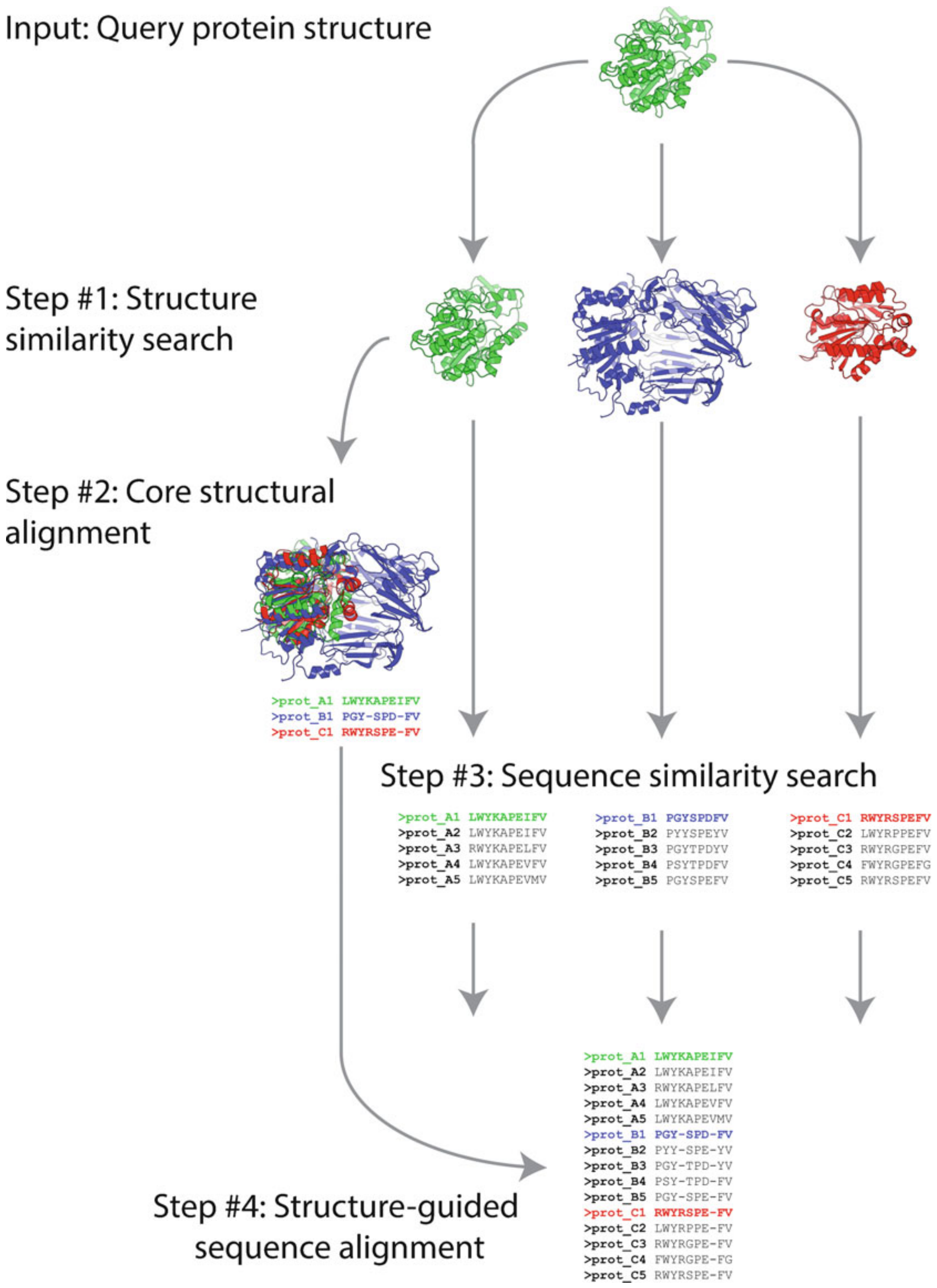


Fig. 1 The outline of the Mustguseal protocol. Reprinted from [3] with permission of the Oxford University Press

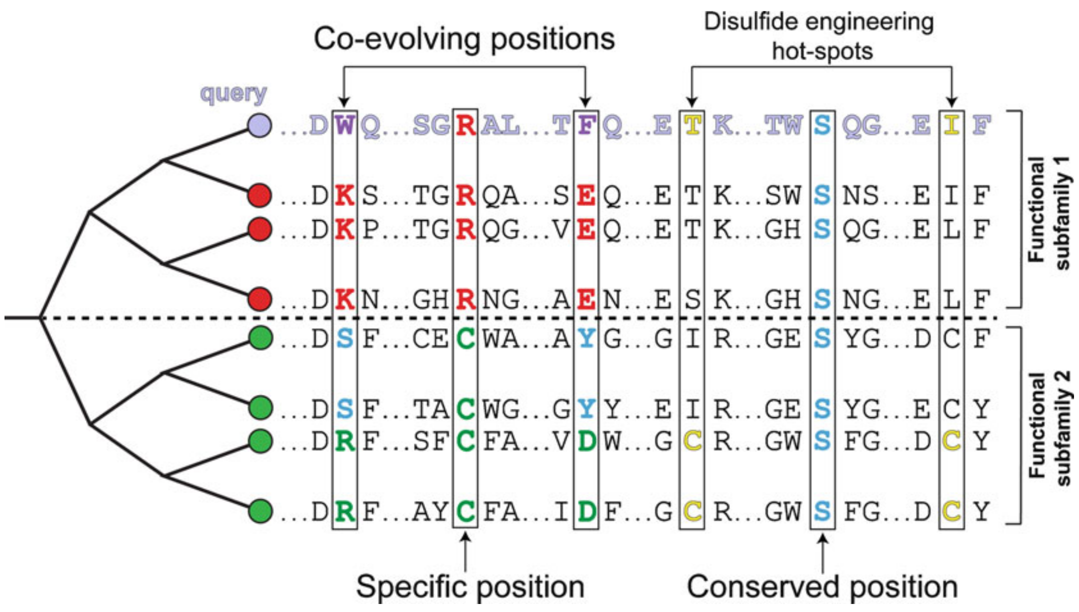

Co-evolving positions
Disulfide engineering hot-spots
query
...DWQ...SGRAL...TFQ...ETK...TWSQG...EIF
...DKS...TGRQA...SEQ...ETK...SWSNS...EIF
...DKP...TGRQG...VEQ...ETK...GHSQG...ELF
...DKN...GHRNG...AEN...ESK...GHSNG...ELF
...DSF...CECWA...AYG...GIR...GESYG...DCF
...DSF...TACWG...GYY...EIR...GESYG...ECY
...DRF...SFCFA...VDW...GCR...GWSFG...DCY
...DRF...AYCFA...IDF...GCR...GWSFG...DCY
Functional subfamily 1
Functional subfamily 2
Specific position
Conserved position

Fig. 2 Patterns of conservation and variability in protein structures that can have implication to their function

should be selected based on the research objective and properties of the particular protein superfamily of interest. In other words, there is no one way to build a multiple alignment, and it has to be specifically constructed taking into account the aim of the bioinformatic analysis.

Collection and subsequent superimposition of proteins is the first step in a pipeline of methods for comparative analysis of homologs. Four sister web-servers are available to further study the alignment created by the Mustguseal and identify patterns of conservation and variability in protein structures that can have implication to their function (Fig. 2). The Zebra web-server can identify conserved and specific amino acid residues [8, 9]. Highly conserved positions seem to appear during the evolution as a result of the selective pressure and therefore are very useful to indicate properties common for the entire superfamily. The specific positions that are conserved only within families/subfamilies, but are different between them, seem to play an important role in functional diversity observed among homologs. Information about both conserved and specific positions can help to understand how the enzyme performs its natural function, while the latter can also be selected as hotspots for directed evolution or rational design experiments in an attempt to improve the wild-type protein variant for a particular purpose [10]. It was also shown that presence of specific positions on the protein surface is a very powerful factor to annotate allosteric sites, which are known to be less conserved and more variable than the catalytic sites [11]. The respective pocketZebra web-server is available to explore the opportunities of the bioinformatic analysis to identify novel regulatory centers in protein structures and study selective accommodation of substrates/inhibitors/effectors. The visualCMAT web-server provides a graphical interface to interpret the role of correlated mutations/co-evolving residues in protein structures, select compensatory mutations for rational design, and study the allosteric communication pathways [12]. Finally, the Yosshi web-server can be used to systematically classify disulfide bonds within the common structural fold of a superfamily and assist to select the most promising hotspots to improve stability of proteins/enzymes or modulate their functions by introducing naturally occurring crosslinks [13]. The bioinformatic and statistical analysis procedures behind these sister web-methods are highly automated within the online implementations, and therefore discussion of the corresponding algorithms is out of the scope of this chapter.

We further provide a step-by-step practical guide to a comprehensive bioinformatic analysis of a protein superfamily using a web-browser as the main tool and notes on selecting the appropriate values for the key algorithm parameters.

2 Materials

Mustguseal and sister web-methods can be operated entirely online via the web-interface. The results are web-based and can be studied on the website using the integrated interactive analysis tools implemented in HTML5. As an option, the results of each web-server can also be downloaded and assessed locally on your computer, what requires some additional software (i.e., a plain text editor, a 3D-structure viewer, etc.). Installation of these supplementary programs does not require significant investment of time from the user.

2.1 Web-Browser

The key software required on the user side to perform the protocol explained in this chapter is an HTML5-compatible web-browser. Current versions of all major browsers (e.g., Google Chrome, Mozilla Firefox, Opera) support HTML5 as the default standard.

2.2 Plain Text Editor

To study the output text files on a local desktop station, you can use a plain text editor, e.g., ConTEXT in MS Windows (http://www.contexteditor.org/) or Kate in OS Linux (the latter usually comes preinstalled).

2.3 3D-Structure Viewer

You can use the PyMol Molecular Graphics System to view the protein PDB structures and the content-rich all-in-one PyMol session files prepared by the Mustguseal and sister web-servers. In OS Linux, PyMol can be installed automatically from a repository (e.g., in openSuSE run the "`zypper in pymol`" command as user with the root privileges) or compiled from sources available at https://github.com/schrodinger/pymol-open-source (recommended for advanced users only). For MS Windows, the unofficial precompiled binaries (i.e., wheel files for the Python installer) are available (find the currently available solution at https://pymolwiki.org/index.php/Windows_Install). The latest PyMol build for your OS can also be purchased from the Schrödinger LLC (http://pymol.org/).

2.4 Sequence Alignment Editor

You may need to use a sequence alignment viewer to manually refine the alignment. We recommend the cross-platform Jalview software [14] (*see* also Chap. 13).

2.5 Perl Interpreter

Optionally, you may want to use Perl to prepare and compile the PyMol session file with a 3D-annotation of the basic alignment statistics on a local desktop station. Most OS Linux distributions have Perl preinstalled. Perl interpreters are also available for OS Windows (e.g., ActiveState Perl). The Perl script used by the Mustguseal to prepare the 3D-annotation PyMol session files can be downloaded from https://mustguseal.belozersky.msu.ru/downloads/annotation_script_latest.tar.gz. This script also requires the MAFFT sequence alignment software [7], which has to be installed separately by the user.

3 Methods

Choose a query protein to represent the protein superfamily of interest based on your particular task and primary interest. It can be the target protein selected for further experimental design, the most studied member of a superfamily, or a protein which you are the most familiar with. You should be able to provide a PDB code of the selected query protein as input to the web-server. The collection and subsequent alignment of protein sequences and structures, starting from this query, is fully automated. However, to improve the accuracy of the alignment and its fidelity to the research objectives, we recommend that you use manual curation at certain steps of the protocol (see below).

In this section, we construct an alignment of a protein superfamily using the Mustguseal for further bioinformatic analysis by the sister web-methods. The aspartate aminotransferase superfamily (i.e., the fold-type I PLP-dependent enzymes), which implements versatile catalytic activities (aldolase, transaminase, decarboxylase, etc.) within a common structural framework, will be used as an example to discuss in details the advanced methodological steps of a recently conducted bioinformatic analysis of this large group of proteins [15]. The PLP-dependent L-allo-threonine aldolase from *A. jandaei* (PDB 3WGB, a homotetramer consisting of four equivalent chains) was selected as the query protein in that study because it was available for the experimental site-directed mutagenesis.

3.1 Define the Diversity and Scope of Your Alignment

The scope of the final alignment constructed by the Mustguseal is defined by the diversity of proteins in the core 3D-structural alignment that represent different families within the superfamily (steps 1 and 2 on Fig. 1). These representative proteins should be carefully selected by implementing the appropriate structure similarity search parameters, as well as exercising manual curation, as explained below.

1. Go to the submission page of the Mustguseal web-server (available at https://biokinet.belozersky.msu.ru/mustguseal).
2. In the "Choose input mode" panel, select the "Mode 1: Submit a query protein."
3. In the "Choose a scenario" panel, select the "Scenario 1: The default." This will set the structure similarity search threshold so that at least 70% of the target from PDB has to make at least 70% of the query protein to be selected for further consideration (*see* **Note 1**).
4. In the "Query protein" panel, enter the PDB code "3WGB" and the chain ID "A."
5. Press the "Submit" button and wait for your task to complete. Then, press the "Results" button to view the results. The "Analysis" page can be used to assess the quality of the produced superimposition online (*see* **Note 2**).
6. Based on the results of the structure similarity search, select representative proteins to define the scope and diversity of your alignment. This step can be performed automatically (see below); however, we recommend manual curation. Download the "Structure similarity search results" package to your local desktop station. Study the file entitled "`superimpose.list`" with the detailed list of all PDB entries selected as being structurally similar to the query protein. For each match, its title, the source organism and the degree of resemblance with the query (i.e., RMSD, sequence identity, number of matching secondary structure elements) are provided. Entries marked by the "*" sign indicate the nonredundant set of proteins characterized by sequence similarity of not more than 95%. Equipped with this information, you should choose at most 32–64 representative proteins based on your particular research objective. Generally speaking, each selected protein should correspond to a family/group with a specific property (e.g., type of the main catalytic reaction/pathway) which is different in other families/groups. It is not recommended to select too many representative proteins (*see* **Note 3**). If, nevertheless, you would need to select more than 32–64 structures for the core alignment, *see* **Note 4**. *See* **Note 5** for other nuances of this step and the automatic selection of the representative proteins by the Mustguseal. *See* **Note 6** for an example.

7. Construct a 3D-structural alignment of the selected representative proteins. This step can be performed automatically (*see* **Note 5**); however, we recommend manual curation. You can construct the respective 3D-superimposition online using one of these web-servers: MATT (http://matt.cs.tufts.edu/) [6], mTM-align (http://yanglab.nankai.edu.cn/mTM-align/) [16], PROMALS3D (http://prodata.swmed.edu/promals3d/) [17], or MAFFT-DASH (https://mafft.cbrc.jp/alignment/server/) [2]. The input to these web-methods is either a text string of PDB codes with chain identifiers, or a pack of PDB files which have to be uploaded to the server from a local computer, or a text file with the corresponding protein sequences. The respective data can be found in the package downloaded from Mustguseal at **step 6** (*see* **Note 5**). Advanced users can run the 3D-alignment on their own resources to speed up the calculations [4]. *See* **Note 7** for an example.
8. Save the file with the core 3D-structural alignment of the representative proteins in the FASTA format (i.e., the sequence representation of the three-dimensional superimposition of homologs) on your local desktop station. Change the file extension to "`.fasta_aln`" if different.

3.2 Enrich the Core 3D-Alignment with Data by Adding Sequences

This section describes further steps to enrich the core 3D-structural alignment of the distant relatives (i.e., that are expected to represent different protein families, steps 1 and 2 in Fig. 1) by adding sequences of their close homologies (i.e., members of these families, steps 3 and 4 in Fig. 1). We suggest two scenarios to set up the parameters depending on your research objectives: to collect a *representative* set of homologs or a *redundant* set of homologs. If your intention is to use the final alignment to study the sequence statistics (e.g., identify conserved, specific, or co-evolving positions in protein structures), very similar sequences should be removed from the alignment. Such a nonredundant (i.e., representative) set of proteins is usually much smaller in size, which facilitates further processing and analysis. On the other hand, even subtle changes in a protein sequence may alter its function. By implementing the redundancy filter, you get a set of proteins that evenly represent the sequence diversity among homologs but may lose these small potentially important differences. Consequently, if you are prepared to study the features of each collected entry individually (e.g., the presence of cysteines potentially capable to form a covalent cross-link in each protein sequence), all sequences currently available in the database should be considered (i.e., the redundant collection). The step-by-step guide is provided below.

1. Go to the submission page of the Mustguseal web-server (available at https://biokinet.belozersky.msu.ru/mustguseal).

2. In the "Choose input mode" panel, select the "Mode 2: Submit a core structural alignment."
3. In the "Core structural alignment" panel, upload the FASTA file corresponding to the core 3D-structural alignment created after applying all steps of Subsection 3.1.
4. Set the sequence similarity search database to the "UniProtKB/Swiss-Prot+TrEMBL" (*see* **Note 8**).
5. Set the sequence length filter threshold to 9999 (%) (*see* **Note 9**).
6. Set the "Dissimilarity filter threshold" to 0.5 bit score per column (*see* **Note 10**).
7. Set the "Use MAPU to build large alignments" to "Yes" (*see* **Note 11**).
8. Set the "Redundancy filter threshold" and "Maximum number of sequences to collect in each subsearch" depending on your research objectives. To construct the *representative* set, define these two parameters as equal to 90% and 500; to construct the *redundant* set—to 100% and 1000, respectively. Instead of manually performing **steps 4–8** of this subsection, you can select "Scenario 2" or "Scenario 3" in the "Choose a scenario" panel, respectively.
9. Press the "Submit" button and wait for your task to complete. Then, press the "Results" button to view the results. *See* **Note 12** for an example of the finally created Mustguseal alignment. The "Analysis" page can be used to assess the quality of the produced superimposition (*see* **Note 2**).

3.3 Further Analysis by the Sister Web-Methods

The finally created multiple alignment incorporates the currently known sequence variability within a common structure of the query protein and can be submitted for further bioinformatic analysis to sister web-servers of the Mustguseal. A new task can be started at the Mustguseal "Results" page (scroll down for section "Advanced Tools to Study the Mustguseal Alignment"). Choose a server depending on your research objectives (i.e., Zebra, pocketZebra, visualCMAT, or Yosshi), and press the respective "Submit to …" button. The multiple alignment will be automatically uploaded from the Mustguseal to the selected web-server, and you will be redirected to a corresponding submission page. Upload the query protein 3D-structure manually (if needed) and press the "Submit" button on that page. This will start the analysis with the default settings suitable in most cases. The results are primarily web-based and viewable online. The web-interface is intuitive and easy to use. We would like to add just a few details.

1. Use the Zebra web-server to identify common and specific features in functionally diverse homologs (i.e., the conserved

and specific positions); use the pocketZebra web-server to annotate novel binding sites in protein structures; use the visualCMAT web-server to identify the correlated mutations/ co-evolving residues; use the Yosshi web-server to classify disulfide bonds within a common fold of the superfamily.

2. The bioinformatic analysis provided by the Yosshi web-server can benefit from the alignment of the *redundant* set of homologs, since each protein sequence is individually evaluated by the algorithm to select potential disulfide bonds. Alignment of the *representative* set of homologs should be used for the analysis by other web-methods.
3. When using the Zebra/pocketZebra, you should start two independent tasks. The first time submit your data with the default settings (i.e., the "Min. size of a subfamily" set to 5% of the alignment size). The second time set the "Min. size of a subfamily" to three proteins (*see* **Note 13**). Study both results.
4. When using the visualCMAT, you can try raising the "Minimum Z-score" parameters (e.g., to 4) to facilitate the study of the most statistically significant co-evolving residue pairs (*see* **Note 14**).
5. When using the Yosshi, you should start two independent tasks. The first time submit your data with the default settings (i.e., the "3D-motif analysis" set to the "Flexible" mode). The second time set the "3D-motif analysis" to the "Rigid" mode (*see* **Note 15**). Study both results.
6. When using the Yosshi to refine the selection of hotspots for protein disulfide engineering in a particular region of the query structure, you can set the "Alignment window" parameter to a nonzero value (e.g., 1 or 3) to consider positions adjacent to those identified by the bioinformatic analysis (*see* **Note 16**).

4 Notes

1. The percentage of secondary structure equivalences between the query and target PDB records is probably the most important parameter in the Mustguseal pipeline, as it defines the selection of representative proteins and, therefore, the scope and diversity of the final alignment. The "70–70%" is a general-purpose pair of thresholds that can be used to collect evolutionarily remote and functionally diverse proteins of comparable dimensions sharing a sufficient enough structure similarity to produce a meaningful superimposition [5]. To include evolutionarily more distant proteins in the alignment for a particular purpose, the user can set the thresholds to lower values, e.g., "50–50%." While doing so, you should keep in mind that

at least some level of similarity may be found in almost any pair of protein structures picked at random from the PDB database. Therefore, decreasing the two thresholds to very low values (e.g., "30–30%") may help to identify all available members of a superfamily, but increases the probability to collect unrelated (i.e., not homologous) proteins which cannot be reasonably compared by global alignment of either sequences or structures. Finally, nonsymmetrical thresholds should be used for superfamilies that contain proteins with significantly different dimensions, domain organization, and/or topology. For example, neuraminidases A (NanA) and B (NanB) from *S. pneumoniae*, sialidase from *T. rangeli*, and sialidase from *V. cholerae* all contain structurally similar catalytic and lectin domains assigned to the GH33 and CBM40 families of the CAZy classification (Fig. 3). However, the number of lectin domains, their sequence content and topology (C- or N-terminal), as well as structural organization regarding the

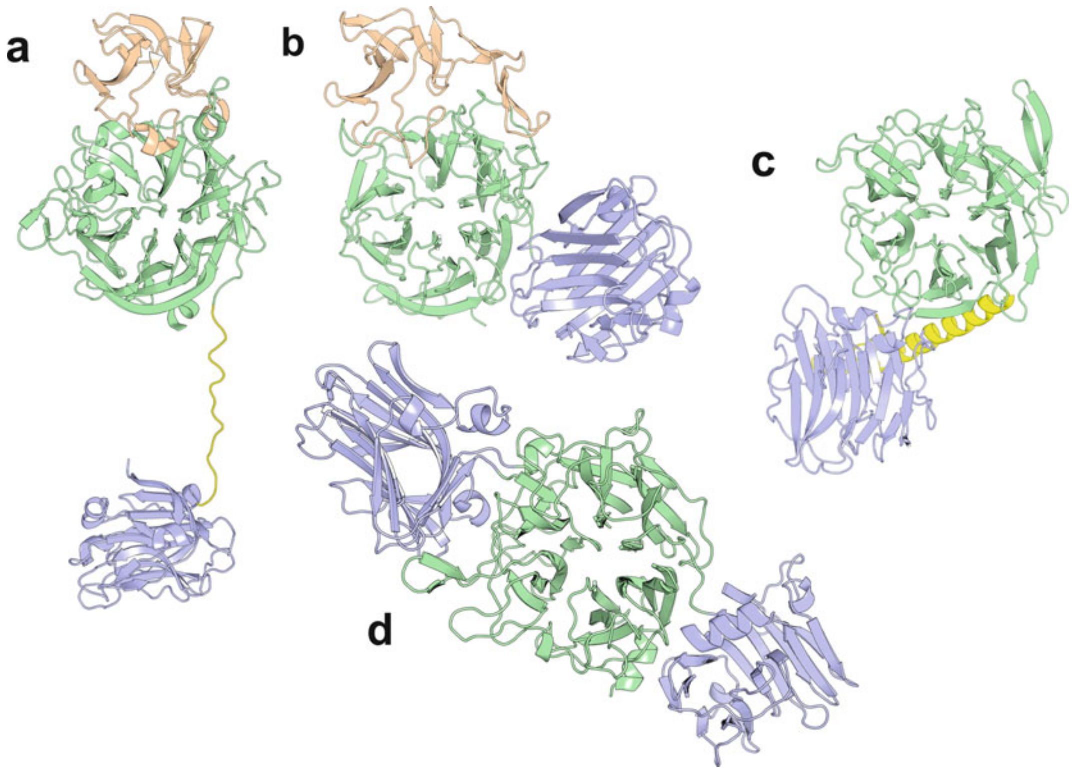


Fig. 3 Modular organization of GH33 catalytic and CBM40 lectin domains in homologous sialidases: (**a**) NanA from *S. pneumoniae*, (**b**) NanB from *S. pneumoniae*, (**c**) sialidase from *T. rangeli*, (**d**) sialidase from *V. cholerae* [18]. Domain color legend: lectin domain (blue), interdomain linker (yellow, where available), catalytic domain (green) with the insertion domain (wheat, where available). The topology of lectin domain is C-terminal in sialidase from *T. rangeli* and N-terminal in other homologs; sialidase from *V. cholerae* additionally features an insertional lectin domain

catalytic domain is significantly different among homologs [18]. To collect all these proteins at once, we suggest using the structure of NanA catalytic domain (available in PDB as a separate entry 2YA8) as a query, setting the structure similarity search thresholds to "60–30%" and then manually reviewing the output [18].

2. At the "Results" page, press the "Analysis" button. The Analysis page offers the basic alignment statistics: the total number of proteins and columns in the alignment; protein length statistics (i.e., the longest, smallest, average); alignment coverage statistics quantified by the amount of columns containing 0%, 5%, 30%, or 50% of gaps compared to the average protein length; and column conservation statistics quantified by the amount of columns containing 100%, 95%, 75%, or 50% of the most frequent amino acid residue. A graphical annotation of the representative protein 3D-structure according to the basic alignment statistics is provided by the interactive 3D-viewer [19] (Fig. 4). Amino acid residues that are at least 95% conserved in the alignment are colored in yellow and shown as sticks. The gradient paint of the protein backbone indicates the degree of sequence conservation in a corresponding position of the multiple alignment, quantified by the Shannon entropy (yellow—highly conserved, gray—variable). Red paint highlights positions in protein structures which are aligned to columns with more than 5% of gaps. These features can help to

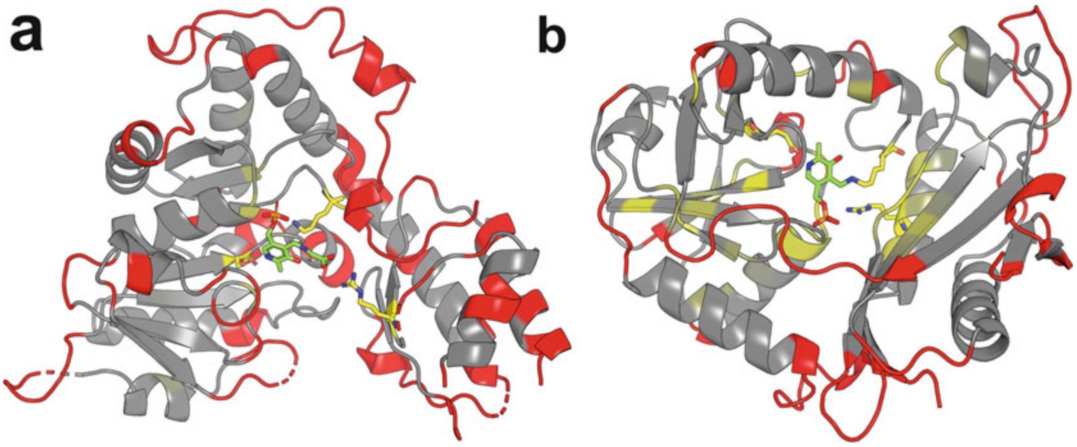


Fig. 4 Examples of the Mustguseal 3D-annotation of the basic alignment statistics: (**a**) annotation of PLP-dependent L-allo-threonine aldolase from *A. jandaei* (fold-type I, PDB 3WGB) according to the alignment statistics calculated from a representative set of 4309 superimposed homologs (qualitatively similar to what was previously discussed in [15]); (**b**) annotation of aminotransferase from *T. uzoniensis* (fold-type IV, PDB 5 CE8) according to the alignment statistics calculated from a representative set of 4143 superimposed homologs. Three catalytically important cofactor-binding amino acid residues are at least 95% conserved in both superfamilies and shown as sticks colored in yellow. The PLP cofactor is colored in chartreuse

assess the quality of your alignment online. Generally speaking, the core structural framework, which usually includes the catalytic sites in enzymes, should have structural equivalences in the majority of homologs (i.e., the key functionally important parts of the structure for the most part should not be colored in red). Loops that are flexible in a protein structure can be colored in red even if they are shared by all homologs due to limitations of the current methods for 3D-structure alignment. In such cases, we recommend to exercise manual curation (i.e., edit the corresponding region in the FASTA file of the core 3D-structural alignment using Jalview). At least 1–3 amino acid residues should be conserved in sequence (i.e., 95% or more) even in a large superfamily. The appropriate coverage depends on the variability of the structural organization and domain composition in the superfamily of interest, but the amount of gapless columns (i.e., with at most 5% of gaps) should be as high as possible for an alignment to be considered informative (we would say, at least 30–50% from the average protein length even in a huge superfamily [20]). If these quality indicators in your alignment are inconsistent with the guidelines explained above, this would either be a special case or (which seems more likely) would indicate a poor quality of the produced superimposition. We recommend checking the core structural alignment and, if necessary, reviewing the choice of the representative protein 3D-structures, followed by a manual curation to improve the corresponding 3D-superimposition. Please note that in the Modes 2 and 3, the graphical annotation of the representative protein 3D-structures according to the alignment statistics is not available online (i.e., only the text version is provided). Instead, a link to the Perl script is available that can be used to create the corresponding 3D-annotation locally on your computer using PyMol. The Analysis page also contains HTML5 plugin that provides a comprehensive toolbox for online analysis of the sequence alignment [21]. In practice, this feature is useful to study only relatively small superimpositions (e.g., up to 1000 proteins) and may become very slow and inconvenient when operating a significantly larger dataset.

3. Why not just select all PDB structures that were found to be similar to the query and construct one very large 3D-structure alignment? In short, because by doing so, you are likely to lose potentially important information. The alignment coverage is a key marker of its value and corresponds to the amount of structural equivalences which are shared by the majority of proteins from the input set. Proteins within large superfamilies are characterized by a significant structural diversity. The more structures are included into the alignment, the smaller are the

conserved regions among all these proteins [4, 22]. The alignment columns outside the common core usually contain a large amount of gaps, as they correspond to regions of the structure that are specific only to some proteins and are absent from the majority of others. These columns overpopulated by gaps are typically dismissed by the bioinformatic software for protein alignment analysis as low-informative and due to limitations of statistical models that assess the significance of amino acid substitutions in homologs. Therefore, the more protein structures you align, the less becomes the alignment itself, quantified by the length of the regions that are shared by all proteins and thus can actually be assessed by the subsequent bioinformatic analysis. Large 3D-structural alignments of protein superfamilies can be of high significance and importance for specific tasks, e.g., to study the mechanisms common among these homologs and promote the development of novel computational techniques to extrapolate functional relationships by a systematic analysis of all available structural data [4]. The "focused" alignments that contain only the selected families portrayed by 3D-structures of the representative proteins (that can still include thousands of protein sequences, i.e., see Subsection 3.2 of this chapter) seem more likely to help in the general case [3, 22].

4. Mustguseal limits the number of representative proteins to at most 16–64 3D-structures in the automatic "Mode 1" and "Mode 2" depending on the particular setup, and to at most 150 3D-structures in the manual "Mode 3," to increase the overall performance of the service and provide it to as many people as possible (see the online documentation for details at https://biokinet.belozersky.msu.ru/mustguseal_limitations). If you follow the protocol as outlined in this chapter (i.e., first construct the core 3D-structural alignment in the "Mode 1" and then submit it for further processing in the "Mode 2") and would like to select more than 64 but at most 150 representative proteins, you would need to use a combination of the "Mode 2" and "Mode 3" to construct your alignment. First, create the core 3D-structural alignment as explained in Subsection 3.1 of this chapter. Then divide this alignment (i.e., the FASTA file) into parts of at most 64 proteins per file. Open the corresponding FASTA file in a text editor, copy the first 64 protein entries (i.e., names and sequences including gaps), and paste them into a separate text file. Then copy the next 64 protein entries into a second file and so on. You may remove the gap-only columns within these sub-alignment files, but otherwise the superimposition of proteins (i.e., the amino acid correspondence by columns and rows) should remain intact. Then, individually process each sub-alignment file in the

"Mode 2" as explained in Subsection 3.2 of this chapter. After each run, download the "Sequence similarity search results" package and save all files of the type `<ID>_<PDB_code>.final.fasta` (i.e., files with the results of the sequence similarity search using the respective PDB file as a query) to a local folder. Change the extension of the collected files to "`.fasta_aln`" and compress them using TAR and GZIP into a "`.tgz`" archive. Finally, submit the original (full) core 3D-structural alignment and the "`.tgz`" file with sequence alignments in the "Mode 3" to merge them together. The technical troubleshooting guide to follow-up on this Note is available at https://biokinet.belozersky.msu.ru/mustguseal-input.

5. For your reference, the content of the "Structure similarity search results" package is explained in details in the online manual available at https://biokinet.belozersky.msu.ru/mustguseal-results. In particular, the PDB files for the 95%-nonredundant set of proteins discovered by the structure similarity search are provided in the `results_nr95_ordered/` folder. It might be helpful to actually examine these structures using a 3D-viewer (e.g., to check for the missing loops or distorted regions in some PDB entries that may be clearly visible in other PDB entries of the same protein or its closely related homolog) to refine you selection of the representative proteins. Alternatively (i.e., instead of **steps 6** and **7** in Subsection 3.1), you can review the representative proteins which were automatically selected by the Mustguseal. The web-server attempts to select at most 32 protein 3D-structures for the core alignment by implementing the redundancy filter at 95, 90%, ..., 40% sequence similarity thresholds (or further below 40% down to 15%, if the corresponding flag was checked on the submission page) [3]. The automatically selected collection of the representative proteins can be downloaded at the "Results" page as the "Core structural alignment" package containing the superimposed PDB structures (in the `aligned_pdbs/` folder) and the sequence version of the 3D-structural alignment (as a FASTA file). If the automatically selected representative proteins comply with your research objective, then **steps 6** and **7** in Subsection 3.1 can be skipped.

6. The aim of a recent study of the fold-type I PLP-dependent enzymes was to identify the amino acid residues that determine the reaction specificity in members of the superfamily [15]. Driven by this challenge, 16 proteins structurally similar to the query 3WGB (i.e., L-allo-threonine aldolase from *A. jandaei* which was available for the experimental site-directed mutagenesis) were selected to represent families

of the aspartate aminotransferase superfamily with different reaction and substrate specificities (i.e., annotated in the PDB as L-allo-threonine aldolase, threonine-phosphate decarboxylase, alanine-glyoxylate aminotransferase, tyrosine aminotransferase, acetylornithine aminotransferase, L-tyrosine decarboxylase, etc.): 3WGB, 2W7E, 3A2B, 1IBJ, 1LKC, 1 V72, 2BKW, 2EZ2, 4IX8, 1QZ9, 4BA5, 4ADE, 3F9T, 4DBC, and 1D2F (chain A in all cases). The missing loop regions in these PDB structures were reconstructed using the MODELLER software [23].

7. In a recent study of the fold-type I PLP-dependent enzymes, the MATT algorithm was used to prepare the initial 3D-structure superimposition of the selected representative proteins (*see* **Note 6**) [15]. This automatically constructed alignment was manually assessed using the Jalview editor and the PyMol to review the pairwise 3D-structural superimpositions of the input proteins. This expert refinement was focused on improving the alignment quality of the flexible loop regions and verifying that the alignment software correctly treated the key catalytic lysine in structures that featured this residue covalently modified by the PLP cofactor.

8. The user can choose between “UniProtKB/Swiss-Prot” and “UniProtKB/Swiss-Prot+TrEMBL” to run the sequence similarity search step within the Mustguseal pipeline (Fig. 1). The Swiss-Prot database contains a relatively small set of ~560 thousand experimentally annotated proteins, while the TrEMBL database features a huge collection of ~182 million mostly unstudied entries. The problem with choosing only the Swiss-Prot for the sequence similarity search is that this collection is biased toward the well-known highly researched proteins. It may not contain homologs of your query (or some of the selected representative proteins), potentially leading to underrepresentation of some families and overrepresentation of the others in the final alignment. On the other hand, the search versus the TrEMBL database is computationally harder and takes a significantly larger amount of time. We recommend choosing the “UniProtKB/Swiss-Prot” option for a quick test run of the Mustguseal pipeline, or when only the structure similarity search and subsequent 3D-structure alignment are of interest. For the “production run,” always select the “UniProtKB/Swiss-Prot+TrEMBL” databases.

9. Each set of sequences collected by the sequence similarity search as a close homolog of the query/representative protein is further filtered by the length to exclude too small and too large sequences (i.e., the “Sequence length filter”). By default, entries that deviate more than 20% from the respective query/representative protein are excluded from further consideration.

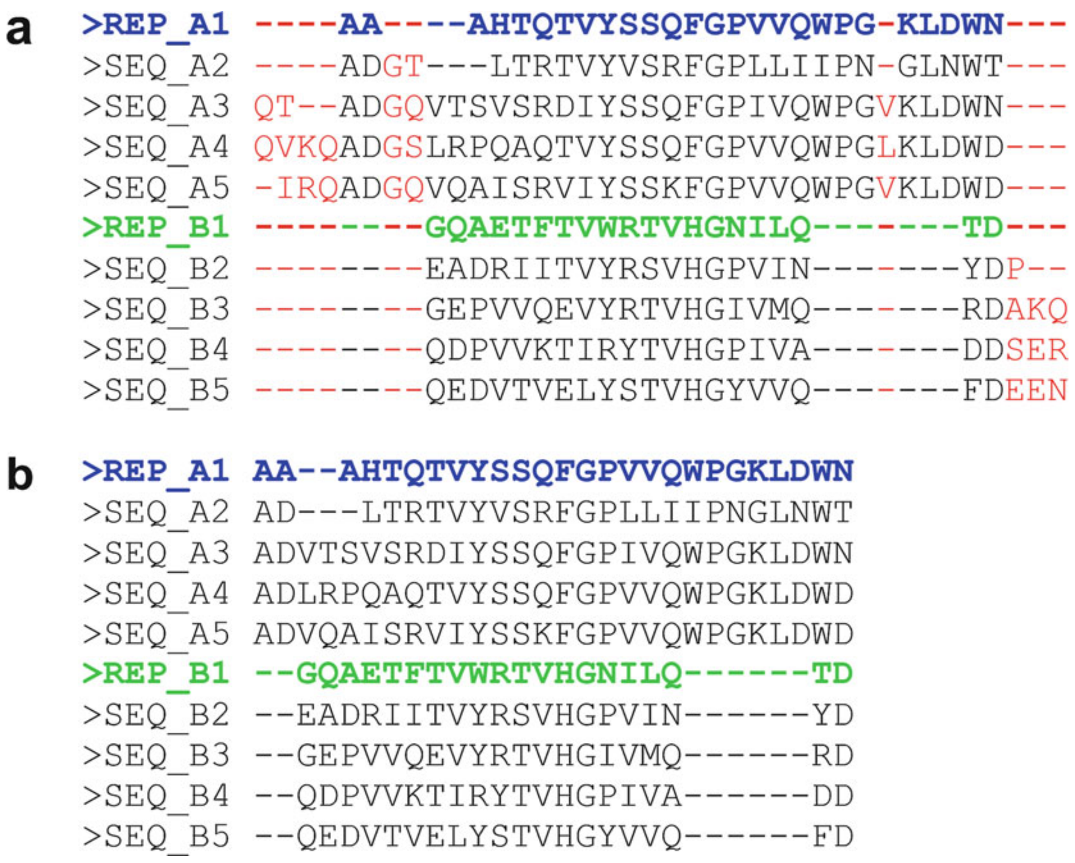
Such outliers usually correspond to incomplete or incorrect database entries, which do not contribute to the alignment value, but significantly increase the number of low-information columns with a high content of gaps. This strategy usually helps to improve the quality of the alignment, optimize the computational efficiency of further processing, and make it more human-readable. However, sometimes it can lead to loss of data. This happens if the protein entry in the PDB database (i.e., a particular protein chain selected as the query) is significantly different in length compared to the corresponding record in the sequence database. For example, the NanA from *S. pneumoniae* contains lectin, catalytic, and membrane-bound domains connected by flexible linkers within a single polypeptide chain of 1035 amino acid residues available in UniProtKB as, e.g., P62575 (Fig. 3a). The same protein is represented in the PDB by two separate entries of only the lectin domain (e.g., 4ZXK containing 188 amino acid residues) and the catalytic domain (e.g., 2YA8 containing 470 amino acid residues) [18]. In a similar example, penicillin acylase from *E. coli* is a heterodimer featuring chains A (209 amino acid residues) and B (557 amino acid residues) within a single globule (e.g., PDB 1GM9). The respective UniProtKB entry P06875 contains both chains covalently connected by a spacer peptide within a single sequence of 846 amino acid residues corresponding to the precursor protein [24]. To include these sequences into the alignment, the sequence length filter should be released. For example, set the threshold to 121% (or to any larger value) to fulfill the case of neuraminidases, as the complete length of the NanA's polypeptide chain is ~221% of the length of the catalytic domain in the PDB 2YA8. For the general case scenario, we recommend to effectively disable the "Sequence length filter" by setting it to 9999%. If nothing interesting is found among the collected sequences with a longer length, you can rebuild the alignment in a separate task with a lower threshold.

10. Each set of sequences collected by the sequence similarity search as a close homolog of the query/representative protein is further filtered to exclude proteins that are too distant (i.e., too different) from this query (i.e., the "Dissimilarity filter"). Such outliers can cause major errors during the sequence alignment, and their elimination is a crucial step of the pipeline [25, 26]. By default, sequences sharing <0.5 bit score per column with the query/representative protein are dismissed from further consideration. It can be considered "safe" to drop this threshold down to 0.25 to collect more sequences, if necessary for a particular purpose [26]. Setting the threshold below 0.25 can help to incorporate more diverse proteins into

the sequence alignment. This may become necessary when the protein superfamily of interest has a limited representation in the PDB database, and therefore, the desired diversity cannot be achieved by the structure similarity search alone. This also may lead to errors in the sequence superimposition and should be carried out with caution and by manual curation.

11. The finally created Mustguseal alignment can feature a large number of low-information columns with a high content of gaps as a result of sequence and structural variability among homologs in a functionally diverse superfamily. Columns overpopulated by gaps are usually discarded by bioinformatic software for the subsequent protein alignment analysis, but can increase the demand for computational resources (including the CPU time and the RAM consumption). The Mustguseal Alignment Postprocess Utility (MAPU) was developed to address this issue. The software is used within the Mustguseal pipeline to automatically remove the least information-rich columns, thereby reducing the overall file size and computational complexity of the subsequent analysis. By default, columns that contain gaps in all representative proteins are removed (Fig. 5). In general, such trimming of the multiple alignment by the structures of representative proteins does not lead to loss of information. Further analysis of the final alignment by the sister web-methods (i.e., Zebra, pocketZebra, visualCMAT, and Yosshi) is focused only on the columns that contain amino acid residues of the representative proteins (i.e., the alignment is mapped onto the 3D-structure of the selected query protein), and this information is identical in the original and trimmed versions of the alignment. The trimming can decrease the file size by up to ~99% and optimize the use of computing resources. We suggest that you always set the "Use MAPU to build large alignments" feature to "Yes," unless the complete sequences of all homologs are needed for a particular purpose. If MAPU is enabled, some limitations of the web-server-based implementation of the Mustguseal protocol are lifted, e.g., you can now set the "Maximum number of sequences to collect in each subsearch" parameter to at most 5000 proteins (instead of 1000 proteins when MAPU is disabled). This provides an opportunity to construct multiple alignments of tens of thousands of proteins for a particular purpose. The MAPU software is also available for download at https://biokinet.belozersky.msu.ru/mustguseal-postprocess.

12. Alignments of the fold-type I PLP-dependent enzymes with a particular focus on L-threonine aldolases were created by the Mustguseal, as described in this chapter, using the August 2019 releases of all databases. The finally created



```
a
>REP_A1 ----AA----AHTQTVYSSQFGPVVQWPG-KLDWN---
>SEQ_A2 ----ADGT---LTRTVYVSRFGPLLIIPN-GLNWT---
>SEQ_A3 QT--ADGQVTSVSRDIYSSQFGPIVQWPGVKLDWN---
>SEQ_A4 QVKQADGSLRPQAQTVYSSQFGPVVQWPGLKLDWD---
>SEQ_A5 -IRQADGQVQAISRVIYSSKFGPVVQWPGVKLDWD---
>REP_B1 --------GQAETFTVWRTVHGNILQ-------TD---
>SEQ_B2 --------EADRIITVYRSVHGPVIN-------YDP--
>SEQ_B3 --------GEPVVQEVYRTVHGIVMQ-------RDAKQ
>SEQ_B4 --------QDPVVKTIRYTVHGPIVA-------DDSER
>SEQ_B5 --------QEDVTVELYSTVHGYVVQ-------FDEEN

b
>REP_A1 AA--AHTQTVYSSQFGPVVQWPGKLDWN
>SEQ_A2 AD---LTRTVYVSRFGPLLIIPNGLNWT
>SEQ_A3 ADVTSVSRDIYSSQFGPIVQWPGKLDWN
>SEQ_A4 ADLRPQAQTVYSSQFGPVVQWPGKLDWD
>SEQ_A5 ADVQAISRVIYSSKFGPVVQWPGKLDWD
>REP_B1 --GQAETFTVWRTVHGNILQ------TD
>SEQ_B2 --EADRIITVYRSVHGPVIN------YD
>SEQ_B3 --GEPVVQEVYRTVHGIVMQ------RD
>SEQ_B4 --QDPVVKTIRYTVHGPIVA------DD
>SEQ_B5 --QEDVTVELYSTVHGYVVQ------FD
```

Fig. 5 Alignment (**a**) before and (**b**) after postprocessing by the MAPU. The sequences of representative proteins (i.e., "REP_A1" and "REP_B1") are shown in color. The alignment columns which contain a gap in both representative proteins are colored in red

superimpositions contained 4309 proteins (the representative set) and 22,672 proteins (the redundant set). Generally speaking, a larger alignment does contain more information, but it is only useful if you know how to extract it. Computationally, larger alignments are harder to study. You should avoid including redundant (i.e., very similar) proteins in the alignment unless there is a reason. For most tasks, you should build an alignment from the representative (i.e., smaller) set of proteins.

13. To identify the specific positions in a multiple alignment (Fig. 2), the input set of proteins has to be classified into subfamilies. The advantage of Zebra is that the algorithm can propose these classifications automatically by graph-based clustering at different fragmentation levels [9]. Each classification is further used to predict the specific positions and estimate their statistical significance. The Zebra algorithm for the prediction of functional subfamilies is, in general, robust to the

selection of parameters. However, we recommend running the utility twice. The first time submit your data with the “Min. size of a subfamily” set to 5% of the alignment size (i.e., the default). This setup tends to classify the dataset into several larger groups. The second time set the “Min. size of a subfamily” to three proteins (i.e., the minimum value). This setup tends to classify the dataset into many smaller groups. Both outputs can help to study the diversity of protein families at different levels of functional hierarchy (see [8] for an example).

14. If too many statistically significant co-evolving pairs of positions were identified by the visualCMAT, you can start new tasks gradually raising the thresholds for both “Minimum Zp” and “Minimum Zc” from the default value of 3.5 up to 4, 4.5, etc. This will eliminate less significant co-evolving pairs to facilitate the study of the most significant ones.

15. The hallmark of the Yosshi method is implementation of the 3D-motif analysis with the “Flexible” statistical model to consider the backbone flexibility for protein disulfide engineering [13]. This approach was shown to outperform the existing tools that select candidate positions for disulfide engineering using strict geometric models trained on covalently connected cysteines and static crystallographic structures of the protein of interest (e.g., Disulfide by Design [27] and MODIP [28]). Nevertheless, we suggest that you also run a second Yosshi task this time selecting the “Rigid” mode for the 3D-motif analysis. This run will identify only those positions in the query protein structure as promising sites for S-S bond engineering, whose geometry is highly similar to the geometry of the known covalently connected cysteine residues. It was shown that the “Rigid” mode is more specific and less sensitive compared to the “Flexible” mode, i.e., it is more likely to identify only the “true” disulfides at a cost of failing to identify some hotspots which may, in fact, form a correct disulfide bond in the query protein assuming both residues are mutated to cysteines [13]. Expert analysis of both outputs can help to improve the selection of the most promising hotspots for further experimental evaluation.

16. The bioinformatic analysis of disulfide connectivity in homologs can help to select the most promising hotspots to introduce an S-S bond into the query protein of interest in attempt to improve its properties. However, sometimes the selected residues already form crucial interactions with the surrounding residues (e.g., hydrogen bonds and salt bridges), and their mutation to cysteines can actually reduce structural stability. In such a case, positions adjacent to those identified by Yosshi can be selected as promising sites for disulfide engineering assuming they do not participate in structurally important

interactions. To evaluate whether an S-S bond could fit into such adjacent positions, the user can set the “Alignment window” parameter to a nonzero value (e.g., 1 or 3). Then, the adjacent positions within the defined window will also be assessed by the 3D-motif analysis.

Acknowledgments

This work was supported by the Russian Foundation for Basic Research grant #18-29-13060 and carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011 [29].

References

1. Suplatov D, Kirilin E, Švedas V (2016) Bioinformatic analysis of protein families to select function-related variable positions. In: Svendsen A (ed) Understanding enzymes. Pan Stanford Publishing, Singapore
2. Rozewicki J, Li S, Amada KM, Standley DM, Katoh K (2019) MAFFT-DASH: integrated protein sequence and structural alignment. Nucleic Acids Res 47(W1):W5–W10
3. Suplatov DA, Kopylov KE, Popova NN, Voevodin VV, Švedas VK (2018) Mustguseal: a server for multiple structure-guided sequence alignment of protein families. Bioinformatics 34(9):1583–1585
4. Shegay MV, Suplatov DA, Popova NN, Švedas VK, Voevodin VV (2019) parMATT: parallel multiple alignment of protein 3D-structures with translations and twists for distributed-memory systems. Bioinformatics 35 (21):4456–4458
5. Krissinel E, Henrick K (2004) Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. Acta Crystallogr D Biol Crystallogr 60 (12):2256–2268
6. Menke M, Berger B, Cowen L (2008) Matt: local flexibility aids protein multiple structure alignment. PLoS Comput Biol 4(1):e10
7. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30(4):772–780
8. Suplatov D, Sharapova Y, Geraseva E, Švedas V (2020) Zebra2: advanced and easy-to-use web-server for bioinformatic analysis of subfamily-specific and conserved positions in diverse protein superfamilies. Nucleic Acids Res 48(W1):W65–W71
9. Suplatov D, Shalaeva D, Kirilin E, Arzhanik V, Švedas V (2014) Bioinformatic analysis of protein families for identification of variable amino acid residues responsible for functional diversity. J Biomol Struct Dyn 32(1):75–87
10. Suplatov D, Voevodin V, Švedas V (2015) Robust enzyme design: bioinformatic tools for improved protein stability. Biotechnol J 10 (3):344–355
11. Suplatov D, Kirilin E, Arbatsky M, Takhaveev V, Švedas V (2014) pocketZebra: a web-server for automated selection and classification of subfamily-specific binding sites by bioinformatic analysis of diverse protein families. Nucleic Acids Res 42(W1): W344–W349
12. Suplatov D, Sharapova Y, Timonina D, Kopylov K, Švedas V (2018) The visualCMAT: a web-server to select and interpret correlated mutations/co-evolving residues in protein families. J Bioinform Comput Biol 16 (02):1840005
13. Suplatov DA, Timonina DS, Sharapova YA, Švedas VK (2019) Yosshi: a web-server for disulfide engineering by bioinformatic analysis of diverse protein families. Nucleic Acids Res 47(W1):W308–W314
14. Waterhouse AM, Procter JB, Martin DM, Clamp M, Barton GJ (2009) Jalview Version 2—a multiple sequence alignment editor and analysis workbench. Bioinformatics 25 (9):1189–1191

15. Fesko K, Suplatov D, Švedas V (2018) Bioinformatic analysis of the fold type I PLP-dependent enzymes reveals determinants of reaction specificity in L-threonine aldolase from Aeromonas jandaei. FEBS Open Bio 8 (6):1013–1028
16. Dong R, Peng Z, Zhang Y, Yang J (2018) mTM-align: an algorithm for fast and accurate multiple protein structure alignment. Bioinformatics 34(10):1719–1725
17. Pei J, Kim BH, Grishin NV (2008) PROMALS3D: a tool for multiple protein sequence and structure alignments. Nucleic Acids Res 36 (7):2295–2300
18. Sharapova Y, Suplatov D, Švedas V (2018) Neuraminidase A from streptococcus pneumoniae has a modular organization of catalytic and lectin domains separated by a flexible linker. FEBS J 285(13):2428–2445
19. Hanson RM, Prilusky J, Renjian Z, Nakane T, Sussman JL (2013) JSmol and the next-generation web-based representation of 3D molecular structure as applied to proteopedia. Isr J Chem 53(3–4):207–216
20. Suplatov D, Sharapova Y, Shegay M, Popova N, Fesko K, Voevodin V, Švedas V (2019) High-performance hybrid computing for bioinformatic analysis of protein superfamilies. In: Voevodin V, Sobolev S (eds) Communications in computer and information science, vol 1129. Springer Nature, Switzerland AG, Basel
21. Gille C, Fähling M, Weyand B, Wieland T, Gille A (2014) Alignment-Annotator web server: rendering and annotating sequence alignments. Nucleic Acids Res 42(W1): W3–W6
22. Steffen-Munsberg F, Vickers C, Kohls H, Land H, Mallin H, Nobili A, Skalden L, van den Bergh T, Joosten HJ, Berglund P, Höhne M, Bornscheuer UT (2015) Bioinformatic analysis of a PLP-dependent enzyme superfamily suitable for biocatalytic applications. Biotechnol Adv 33(5):566–604
23. Webb B, Sali A (2017) Protein structure modeling with modeller. In: Kaufmann M, Klinger C, Savelsbergh A (eds) Functional genomics. Methods in molecular biology, vol 1654. Humana Press, New York
24. Suplatov D, Panin N, Kirilin E, Shcherbakova T, Kudryavtsev P, Švedas V (2014) Computational design of a pH stable enzyme: understanding molecular mechanism of penicillin acylase's adaptation to alkaline conditions. PLoS One 9(6):e100643
25. Söding J, Biegert A, Lupas AN (2005) The HHpred interactive server for protein homology detection and structure prediction. Nucleic Acids Res 33(Suppl. 2):W244–W248
26. Fischer JD, Mayer CE, Söding J (2008) Prediction of protein functional residues from sequence by probability density estimation. Bioinformatics 24(5):613–620
27. Craig DB, Dombkowski AA (2013) Disulfide by Design 2.0: a web-based tool for disulfide engineering in proteins. BMC Bioinformatics 14(1):346
28. Dani VS, Ramakrishnan C, Varadarajan R (2003) MODIP revisited: re-evaluation and refinement of an automated procedure for modeling of disulfide bonds in proteins. Protein Eng 16(3):187–193
29. Sadovnichy V, Tikhonravov A, Voevodin V, Opanasenko V (2017) "Lomonosov": supercomputing at Moscow State University. In: Vetter JS (ed) Contemporary high performance computing. Chapman and Hall/CRC, New York

Part III

Visualization

Chapter 13

Alignment of Biological Sequences with Jalview

James B. Procter, G. Mungo Carstairs, Ben Soares, Kira Mourão, T. Charles Ofoegbu, Daniel Barton, Lauren Lui, Anne Menard, Natasha Sherstnev, David Roldan-Martinez, Suzanne Duce, David M. A. Martin, and Geoffrey J. Barton

Abstract

In this chapter, we introduce core functionality of the Jalview interactive platform for the creation, analysis, and publication of multiple sequence alignments. A workflow is described based on Jalview's core functions: from data import to figure generation, including import of alignment reliability scores from T-Coffee and use of Jalview from the command line. The accompanying notes provide background information on the underlying methods and discuss additional options for working with Jalview to perform multiple sequence alignment, functional site analysis, and publication of alignments on the web.

Key words Multiple sequence alignment visualization, Interactive analysis, Web application, Desktop application, Functional site inference, Web services

1 Introduction

The Jalview [1] platform has many features for sequence analysis and visualization and is freely available both as a native "app" and single-page web application [2] from its web site [3]. A core function is to make it easy to run state-of-the-art methods for multiple sequence alignment (MSA). The resulting alignments can be visualized and integrated with other information to further interpret them and create figures for publication. In this chapter, we describe the steps involved in a typical Jalview sequence alignment workflow, as depicted in Fig. 1. Sequences for alignment may be retrieved from public databases or loaded via a variety of common file formats. Access to a range of alignment programs is achieved directly within Jalview through web services [4, 5], but Jalview also allows alignments generated by external programs to be imported. Built-in analysis routines calculate the consensus for each alignment column and for proteins, the amino acid physicochemical

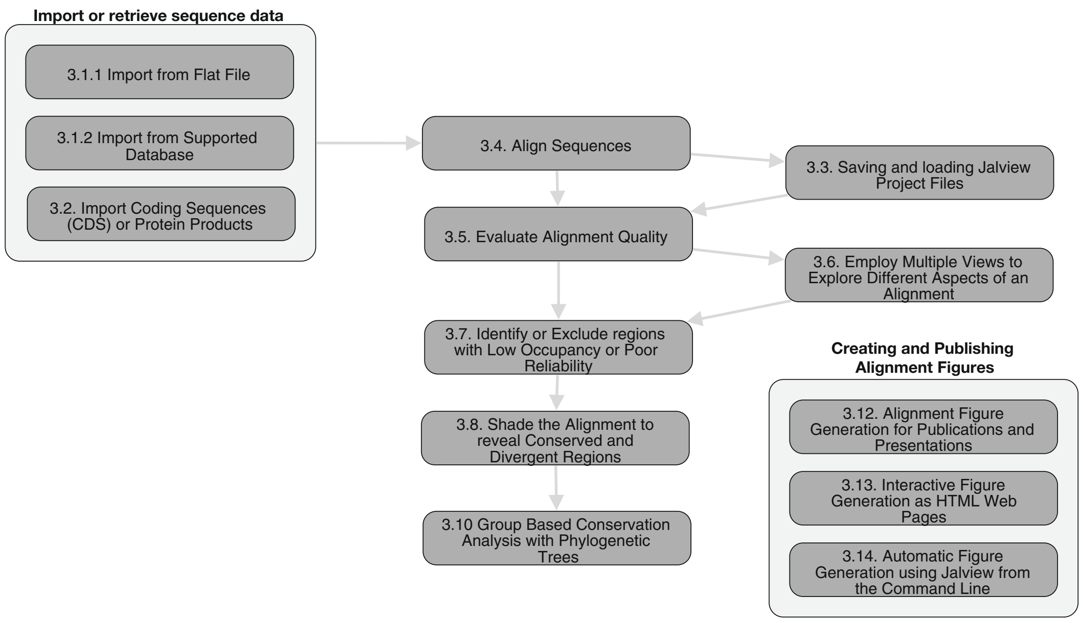

Import or retrieve sequence data
3.1.1 Import from Flat File
3.1.2 Import from Supported Database
3.2. Import Coding Sequences (CDS) or Protein Products
3.4. Align Sequences
3.3. Saving and loading Jalview Project Files
3.5. Evaluate Alignment Quality
3.6. Employ Multiple Views to Explore Different Aspects of an Alignment
3.7. Identify or Exclude regions with Low Occupancy or Poor Reliability
3.8. Shade the Alignment to reveal Conserved and Divergent Regions
3.10 Group Based Conservation Analysis with Phylogenetic Trees
Creating and Publishing Alignment Figures
3.12. Alignment Figure Generation for Publications and Presentations
3.13. Interactive Figure Generation as HTML Web Pages
3.14. Automatic Figure Generation using Jalview from the Command Line

Fig. 1 Workflow for creating, analyzing, and generating figures for a multiple sequence alignment in Jalview. Numbers for each stage correspond to stages described in Sect. 3. An interactive version of this workflow can be viewed online at https:/tess.elixir-europe.org/workflows/multiple-sequence-alignment-analysis-and-figure-generation-with-jalview

properties. These create column Annotations that are shown as histograms under the alignment. The alignment may be filtered to remove redundant sequences and to hide columns with low numbers of aligned residues. Aligned sequences may also be grouped either manually or by partitioning an associated tree which can be imported in a variety of formats or calculated using Jalview's own tree algorithms. A range of shading and coloring schemes allow common patterns to be highlighted according to standard properties for amino acids (such as hydrophobicity), or whether nucleotides are purine or pyrimidine derivatives. Uniquely, coloring may be combined with conservation and consensus calculations to emphasize patterns of variation among conserved regions and to highlight different patterns of conservation among subgroups of sequences in the alignment.

2 Materials

Availability, download, and installation. The latest version of the Jalview Desktop application can be obtained from http://www.jalview.org/download. Installers are provided for ×86-based Windows, Linux, and OSX operating systems, and the minimum recommended physical memory is 1 GB. The installers provide the “Jalview Native Application”—which offers enhanced usability through features such as file-type associations, automatically managed Java installation, and “Over the Air Update” to ensure the latest and most secure version of Jalview is launched. An “executable Jar” file is also provided for users wishing to launch a specific version of Jalview or execute it on a platform for which a Jalview Native Application is not yet available. Jalview packages may also be obtained in Homebrew [6] and BioConda [7] and are primarily designed for users integrating Jalview into a command-line workflow rather than performing interactive analyses as described here.

Memory settings and working with larger datasets. The Jalview Native Application and Jalview Executable Jar will automatically request up to 90% of physical memory to be allocated to a Jalview session. The percentage of memory allocated can be modified via a command-line argument. Currently, no support is provided for working with alignments too big to load into memory.

The JalviewJS Web Application. An alternative to installing the Jalview Desktop is to access the JavaScript version of Jalview at https://www.jalview.org/jalview-js/JalviewJS.shtml. It is designed to work with modern web browsers such as Google Chrome or Mozilla Firefox, and provides the majority of functionality described below, except for access to public web services provided by JABAWS.

3 Methods

3.1 Import or Retrieve Sequence Data

1. A range of common bioinformatics sequence file formats (*see* **Note 1**) can be imported by “drag and drop” or via options provided on the “File” menu. Data from the system clipboard may also be pasted by right-clicking on the desktop background and selecting the “Paste to New Window” menu option that appears.
2. In addition, Jalview provides a “Sequence Fetcher” which allows import of sequences, alignments, and 3D structures from databases hosted by EMBL-EBI [8, 9] (*see* **Note 2**).

Once the import has completed, an alignment window containing the sequence data is displayed. More sequences may be added by the same methods (via the alignment window’s own “File” menu, by pasting from the clipboard, or by dragging files onto the window).

3.2 Importing Coding Sequences (CDS) or Protein Products for CDS

CDS for proteins shown in an alignment view can be added simply by dragging and dropping the file containing CDS onto the protein alignment. Protein products for CDS are loaded in the same way. Providing each CDS has the same name as its corresponding protein product, and codons from the CDS exactly match the amino acid sequence (under one of Jalview's supported translation tables), Jalview will offer the option of opening a "Linked Protein and CDS View" (*see* **Note 3**). This configuration allows proteins to be multiply-aligned according to their amino acid sequence, but analyzed at both the nucleotide and amino acid sequence levels.

3.3 Saving and Loading Project Files

A Jalview Project File (JVP) can be created via the Desktop's File- > Save Project menu entry or an alignment window's File- > Save As... menu entry. JVP files store data and visualization settings for alignment windows, including any associated views such as trees and 3D structures (e.g., the example file that is automatically loaded by default on startup—*see* **Note 4**). It is recommended that work is saved after each stage below to avoid data loss, and if the same filename is used, then Jalview will create versioned backups (*see* **Note 5**).

3.4 Align Sequences

A range of alignment programs is offered via the Alignment submenu of the Alignment Window's "Web Services" menu. Jalview provides access to public services provided by the University of Dundee which are suitable for aligning up to 1000 sequences of at most 1000 residues each, and jobs are allowed to run for up to 1 h. **Note 6** discusses how to perform alignments of larger sequence sets with Jalview.

1. For some alignment programs, Jalview provides a "Realignment" option. This allows sequences to be added to an existing alignment. *See* **Note 7** for a discussion of its effective use.
2. If a selected region is defined in the alignment window (*see* **Note 8**), then only data in that region will be submitted for alignment. Similarly, hidden sequences will not be included in the alignment and hidden columns will force a series of local multiple alignments to be performed on just the visible regions (*see* **Note 9**).

Jalview *must be left running* for the duration of the alignment procedure. For current versions of Jalview (2.11), there is no way to "reconnect" to an alignment or analysis procedure performed via the "Web Services" menu, and Jalview projects do not preserve any information about Web Service jobs that have completed or are currently in progress.

3.5 Evaluating Alignment Quality

Jalview provides visual analytics that help judge the degree of similarity among sequences in an aligned region, but it does not on its own compute measures that indicate how reliable all or part of an alignment may be. Such calculations are provided by the T-Coffee suite via M-COFFEE [10] or Transitive Consistency Score (TCS) analysis [11] (http://www.tcoffee.org/Projects/tcs/) (*see* **Note 10** and Chapter 6). Below we briefly outline a protocol for manually performing an alignment reliability calculation in T-Coffee.

1. Export the alignment to be assessed as a FASTA file via the File- > Save As... menu entry from the alignment window.
2. Submit the FASTA file for assessment with T-Coffee.

   Either via the command line

   t_coffee -infile *prot.aln* -evaluate -output score_ascii.

   or, alternatively, via the T-Coffee web server's TCS submission form at http://tcoffee.crg.cat/apps/tcoffee/do:core
3. T-Coffee's TCS analysis produces a "score_ascii" file which can be read by Jalview. If the web service was employed, then this file should first be downloaded. To view the results, simply drag and drop the file onto the alignment window, or load it via the File- > Load Annotations option.
4. Jalview provides a "T-Coffee Score" color scheme in its Colour menu for viewing the reliability scores produced by TCS. This is enabled by default when the score_ascii file is loaded and mimics T-Coffee's standard TCS coloring: where red indicates the most reliably aligned regions, transitioning to green, yellow, and blue for poor quality regions.

3.6 Employing Multiple Views to Explore Different Aspects of an Alignment

Alignment Views allow different regions of the alignment to be shown or hidden and sequences to be independently grouped and colored. Operations on the alignment, sequence, and annotation data affect all Views. When preparing an alignment figure for inclusion in a publication, it is often useful to create a series of Views corresponding to each panel of the figure.

1. A new View for the alignment is created with the New View option in the View menu. Its presence is indicated by a new tab appearing above the alignment ruler.
2. Views may be displayed simultaneously via the Expand option in the View menu. The Gather option returns all Views to the tab bar on the current View's alignment window.
3. The current View is removed by pressing "Control" or "CMD" and "W" (or if the Views are expanded to their own window, by simply closing that View's window).

Multiple Views allow specific features of an alignment to be highlighted. Accordingly, a View may be given a unique name via the dialog box opened by right-clicking the View's tab.

3.7 Identification or Exclusion of Regions with Low Occupancy or Poor Reliability

Occupancy measures the number of sequences aligned at each position and is one of the dynamic Annotation rows automatically computed for an alignment (*see* **Note 11**).

1. Select the Select/Hide Columns dialog from the Alignment Window's "Select" menu.

2. Choose "Occupancy" from the first drop-down menu as the annotation row to be queried.
3. Choose "Below" from the "Threshold Type" drop-down menu, and check the "Percentage" checkbox. Enter "10" to select columns in the alignment where less than 10% of sequences are aligned.
4. To hide the columns selected by the filter, then select the "Hide" option in the dialog rather than the "Select" option.
5. Press OK to close the dialog, or cancel to reset column visibility.

To exclude unreliable regions according to T-Coffee TCS scores, select "T-COFFEE" from the list of annotation rows in Sect. 3.5, **step 2**.

3.8 Shading the Alignment to Reveal Conserved and Divergent Regions

Jalview has a range of protein and nucleotide color schemes (*see* **Note 12**) which can be applied in combination with values from the Consensus (and for proteins, Conservation and Quality) dynamic Annotation rows (*see* **Note 11**) to highlight variation in columns that exhibit a high degree of amino acid or nucleotide conservation.

1. Revealing conservation patterns with Colour by Conservation (Proteins only).
   (a) Select "Blosum 62" from the Colour menu, which colors each residue on a scale from white to blue according to the likelihood of mutation from the reference or consensus sequence for the alignment view.
   (b) Enable "Colour by Conservation." Columns that exhibit a high physicochemical property conservation score (*see* **Note 11**) will appear more strongly colored than those with fewer conserved properties.
2. Revealing regions with high percentage identity.
   (a) Select "Percent Identity." This scheme shades nucleotides and amino acids according to their abundance at each column in the alignment.
   (b) To shade the alignment according to some other property (e.g., Purines and Pyrimidines, or Taylor's physicochemical property-based color scheme), first apply this scheme, and then select "Above Identity Threshold" from the "Colour" menu to only color symbols present in more than the specified minimum percentage of aligned sequences at each column in the alignment.

3.9 Shading the Alignment According to Conservation Scores from the AACon Web Service

1. Select the "Change AAcon settings..." option from the Conservation submenu of the Web Services drop-down menu in the alignment window.
2. Choose which scores to calculate from the dialog and press "Submit" to enable the calculation and display of AACon annotations. After a short delay, the additional scores will appear as histograms below the alignment.
3. Open the "Colour by Annotation" dialog from the Color menu. Choose one of the annotation rows from the drop-down menu in the dialog. By default, a linear shading will be applied to columns of the alignment according to values in the chosen row. To apply a threshold (similar to the "Percent Identity" threshold above), select the "Use Original Colours" option and select a threshold type from the drop-down menu.

3.10 Group-Based Conservation Analysis with Phylogenetic Trees

Shading and filtering according to column statistics do not always reveal regions of similarity or divergence not shared by all sequences in the alignment (*see* **Note 13**). Once sequences have been grouped, however, the shading schemes introduced in Sect. 3.8 will reveal the patterns of conservation and divergence unique to each group. Jalview is able to create groups interactively from selections and also to subdivide a selection to group sequences according to their identity. However, the most powerful sources of groups are phylogenetic trees (*see* **Note 14**). Jalview can import existing trees or calculate one for aligned sequences. The built-in tree viewer then offers a way to subdivide aligned sequences into groups according to their relatedness as defined by the tree.

1. A tree for all sequences or a selected region of a view can be calculated and displayed via the "Calculate" dialog (accessed via the Calculations menu). For best results when computing subgroups, particularly for a selected region of an alignment, we recommend selecting the UPGMA Average Distance Tree.
2. Use the "Sort alignment by tree" option in the Tree viewer's submenu to reorder sequences in the alignment view according to the tree (Jalview's preferences allow this action to be configured to be performed automatically).
3. Select a position between the root and leaves of the displayed tree to define a set of groups on the alignment. (Warning: previously defined groups will be removed.) In the alignment view, the names of sequences in the same group will have a similar background color. For each group, coloring based on Conservation and Percent Identity will employ values computed for just the grouped sequences: locally conserved regions will therefore be more strongly colored in comparison to regions of local divergence.

4. For alignments too large to fit on screen, the Alignment Overview (opened via the option in the View menu) allows differences between groups to be compared more easily.
5. When the overview is open, "Colour by Sequence ID" can be applied in combination with "Colour by Conservation" or "Above Identity Threshold" to more easily distinguish sequences in different groups.

3.11 Visualizing Group Conservation and Consensus

The "Autocalculated annotation" submenu in the Annotations menu provides options controlling the display of conservation and consensus rows for the currently selected group. To modify settings for all groups, enable the "Apply to all groups" option.

1. The consensus (and for proteins—conservation) annotation for groups on the alignment can be shown by enabling the "Group Consensus" and "Group Conservation" options under the Annotations menu's Autocalculated Annotation submenu.
2. Clicking the label of a group's annotation row will highlight the sequences in that group. Double clicking the label will select the whole group.
3. Display of sequence logos.

   Amino acid and nucleotide distributions may be visualized for an alignment or sequence group as sequence logos [12]—where for each column, letters are stacked in order of increased frequency of observation, and their height also scaled accordingly.
   (a) Select the Annotation menu's Show Logo option to display sequence logos on group or alignment consensus annotation rows.
   (b) Distributions across different sites of the alignment can be more easily compared by enabling the "Normalise Logo" option. In this case, disabling the "Show Histogram" option allows the logo to be more clearly viewed.

3.12 Alignment Figure Generation for Presentations and Papers

Alignment views can be exported in a range of ways via the File menu's "Export" submenu. Views can be exported as shown, rendered as a Portable Network Graphic (PNG) raster image—suitable for onscreen display and Scalable Vector Graphic (SVG) or Encapsulated PostScript (EPS) format vector graphic drawings which are recommended when preparing figures for publication. HTML pages can also be generated via the options in the menu—these interactive export options are discussed in the next step.

3.12.1 Preparing for Figure Export

When preparing to export views as static figures, it is recommended that a new View is created to allow layout, font size, colors, and data visibility to be configured, since no "Undo" functionality is provided to revert changes.

3.12.2 "Wrap mode": Formatting Alignments to Fit Within the Margins of a Page

Enable the Wrap Mode option in the "Layout" menu to format the MSA as a series of fixed width blocks. The number of columns shown in each block is defined by the width of the alignment view. These are reflected by the numberings shown in the alignment ruler. The number of columns may be changed by:

(a) Adjusting the width of the alignment view window. How this is done depends on which operating system (OS) is used: On Mac OSX, move the mouse pointer to the bottom right corner of the window and click-drag to adjust the size. Other OSes allow resizing by click-dragging any edge of the window.

(b) Adjusting font size and column width. Either via the Layout's Font dialog, or with a three-button mouse by clicking the middle button in the alignment view and moving it left or right to adjust width, and up or down to decrease or increase font size (respectively).

(c) Adjusting the sequence ID margin. When the mouse pointer is moved to the right-hand side of the sequence ID panel, it will change to indicate that the margin can be adjusted by click-dragging to the left and right.

3.12.3 EPS Export as "Characters" or Line Art

When EPS export is selected, Jalview can either represent each sequence symbol as a character or render the shape of each symbol in the EPS file. The former allows EPS files that can be easily edited in a vector graphics program such as Illustrator—e.g., to modify sequence ID labels, but can result in EPS files that appear different to Jalview's alignment view. The latter results in larger files but ensures all aspects of the MSA visualization (e.g., sequence logos, character alignment) are faithfully reproduced.

3.13 Interactive Figure Export in HTML Web Pages

Other options in the Export submenu of the Alignment View's File menu allow HTML pages to be generated containing either an embedded SVG rendering of the view or a JavaScript visualization such as the BioJS Multiple Sequence Alignment Viewer (BioJS-msaviewer) [13]. In general, interactive figure export results in web pages that will look and behave differently to visualizations provided by the Jalview Desktop app or the JalviewJS web component (*see* **Note 15**).

3.13.1 Export as an Interactive HTML Figure

The "HTML" export option produces a web page containing two SVG figures—one for the ID panel and one for the columns of the MSA and Annotation rows. Buttons are provided to open the exported view in the Jalview application, and view the embedded data (stored as BioJSON [14]).

3.13.2 Export Alignment for Visualization with BioJS-msaviewer

(a) The “BioJS” export option generates an HTML page containing embedded BioJSON and the JavaScript code necessary to display the alignment with BioJS-msaviewer.

HTML files generated by Jalview that contain BioJSON can be imported like any other alignment file (*see* **step** 3.1). However, Jalview Project files are recommended for long-term archiving of data and visualizations generated during multiple sequence alignment and analysis.

3.14 Automated Alignment Figure Generation in Batch Mode

Jalview’s command line allows figures to be generated without user intervention.

3.14.1 Prepare a Custom Jalview Properties File

When running in batch mode, a custom preferences file allows alignment layout parameters to be specified. To do this:

(a) Make a backup of your existing “.jalview_properties” file.

(b) Adjust the various Jalview User preferences provided in the Visual, Colour, Output, and Editing panels so that when an alignment is imported, it has the desired appearance.

(c) Make a copy of your customized “.jalview_properties” file using a unique name such as “jalview_batch.properties.”

3.14.2 Running Jalview as a Command-Line Program

The precise way that Jalview is called from the command line depends on how it was installed. The procedure below assumes you have downloaded the Jalview executable JAR and that your system has an existing Java 8 installation.

(a) Execute the following command to generate a figure from the command line:

Java -jar jalview-2.11.0.jar -headless -props jalview_batch.properties -open “http://www.jalview.org/examples/uniref50.fa” -png example_fig.png

(b) Verify that the generated PNG file: “example_fig.png” exists and has the desired appearance.

4 Notes

1. Table 1 lists the file types currently supported by Jalview, the type of data provided, and whether Jalview can export as well as import in that format, along with any caveats regarding their use. For completeness, we also include here formats for annotation, 3D structure, phylogenetic trees, and the Jalview specific formats: Jalview Features Format, Jalview Annotations Format, Jalview Project, and BioJSON.

Table 1
File formats supported by Jalview

Format	Extension	Sequence	Features	Annotation	Import only	Notes
AMSA	.amsa	✓		✓	✓	
FASTA	.fa, .fasta	✓				Jalview does not currently support "FASTQ" style quality data embedded in FASTA records
Block	.blc	✓		✓		Header lines are available via the "Alignment Properties" dialog. For BLC files produced by an iterative search program such as ScanPS, the first iteration is loaded by default. A specific iteration can be retrieved by appending the filename with "#2" (to retrieve iteration 2)
Clustal	.aln	✓				The "Clustal" version header in files written by Jalview is hard coded, which may cause compatibility problems with other programs that require a specific version header
MSF	.pileup, .aln	✓				
Protein Information Record	.pir	✓		✓		Jalview supports MODELLER [22] style PIR description lines [23] to specify the filename and chain in a PDB [24] or mmCIF [25] file that maps to a sequence
Phylip		✓				
Pfam	.pfam	✓				
Stockholm	.stk, .stockholm	✓	✓	✓		In addition to sequence, features, and annotation, Jalview also imports any provided database cross-references for a sequence (including PDB [24] database identifiers) and generic annotation tags associated with the alignment
Jalview Features Format	.jvfeats		✓			

(continued)

Table 1 (continued)

Format	Extension	Sequence	Features	Annotation	Import only	Notes
T-COFFEE Score file	.score_ascii			✓		Jalview will raise an error if the number of sequences and columns in the score file does not match the alignment
Jalview Annotations Format	.jvannot			✓		
Generic Features Format (V3)	.gff, .gff2, .gff3	✓	✓			
Variant Call Format (indexed)	.vcf, .vcf.tbi		✓			A protein or DNA contig with chromosomal coordinates must first be loaded before it can be annotated with variants from VCF files
BioJSON	.json, biojson, html	✓	✓	✓		A JSON hash containing elements structured according to BioJSON scheme can also store Groups, color schemes, Hidden columns, and sequences. Jalview will also search HTML files for embedded BioJSON
Jalview Project	.jvp, .jar	✓	✓	✓		Projects preserve all aspects of a Jalview session, including the layout of windows showing alignments, trees, and 3D structures
PDB	.pdb, .ent	✓	✓	✓	*	Only 3D structures imported as PDB [24] files can be exported as PDB files
mmCIF	.mmcif, .cif	✓	✓	✓	*	Only 3D structures imported as mmCIF files can be exported as mmCIF files
Newick	.tree, .nw				*	Jalview supports both simple Newick [26] and Extended Newick [27] and will preserve distances and bootstrap values detected in the file. It only exports trees in simple Newick format

When importing sequence data, Jalview employs a series of rules to determine the format. If no format is found to match based on these rules or an error is encountered while parsing the data according to the determined format, then Jalview will default to the "Pfam" format reader. The Pfam format is also used by default when sequence data are copied directly from an alignment window to the system clipboard for other applications.

2. Table 2 details the databases that Jalview 2.11 is able to access and the kinds of data they provide. Jalview will present an interactive query dialog for databases that support free text search or a simple "Fetch IDs" dialog for databases for which no query client is available. The specific search capabilities provided when performing a free text search depend on the particular database being queried: all provide a range of fields that can be used to restrict a query (via a drop-down menu to the left of the search box) and also allow structured queries to be entered directly (please *see* Jalview documentation for details of these).
3. Jalview allows alignments of proteins and their CDS to be visualized and interactively analyzed as a pair of linked alignment views shown docked bottom to top (*see* Fig. 2). Operations on protein sequences are mirrored on the CDS, allowing proteins to be aligned using their amino acid sequence and the resultant CDS multiple sequence alignment analyzed to investigate the presence of bias that might indicate selection. All analysis steps described above can be performed on a linked CDS/Protein view, with the added benefits that (1) codon bias and diversity can also be visualized and used for filtering the alignment, (2) genome- and transcript-level sequence features can be visualized via the CDS alignment view, and (3) phylogenetic trees computed using the CDS alignment using either Jalview's built-in score models or via an external program and loaded back onto the alignment can provide additional evolutionary insight when used to partition the protein alignment.
4. When first launched, Jalview will automatically import and display an example alignment, tree, and 3D structure retrieved from the Jalview web site. The display of these examples is disabled by opening Jalview's user preferences dialog (via the Preferences option of the "Tools" drop-down menu), un-ticking the "Open File" checkbox, and selecting "OK" to save the updated preferences. Jalview user preferences are stored in a ".jalview_properties" located in the user's home directory. These are read every time Jalview is launched, and a customized properties file can also be provided when Jalview is run in batch mode from the command line to specify alignment and annotation layout for automatic generation of figures.

Table 2
Sequence databases that Jalview can query and retrieve data

Database	Free text search	What is retrieved	
UniProt	Yes	Annotated protein sequences	Imported sequences include positional features and cross-references to other databases
Ensembl		Annotated genomic loci, transcripts, and protein products	Retrieved transcripts are shown aligned against their parent loci. Intronic regions are automatically hidden
European Nucleotide Archive		Annotated contigs and protein products	Retrieved contigs that include CDS regions are shown along with retrieved protein products in a linked CDS/Protein alignment view
PDBe	Yes	3D structures (as mmCIF)	Sequences for all chains in the imported 3D structure file are extracted and shown in the alignment view. Sequence-associated Annotation rows and features provide information about secondary structure and mapping between sequence and 3D coordinate numbering schemes
Pfam		Annotated protein domain alignment (as Stockholm)	Alignments include an additional consensus line computed by the Pfam pipeline and a secondary structure annotation row when 3D structure data for the domain family is available.
Rfam		Annotated RNA domain alignment (as Stockholm)	A consensus RNA secondary structure annotation for the Rfam family is also provided

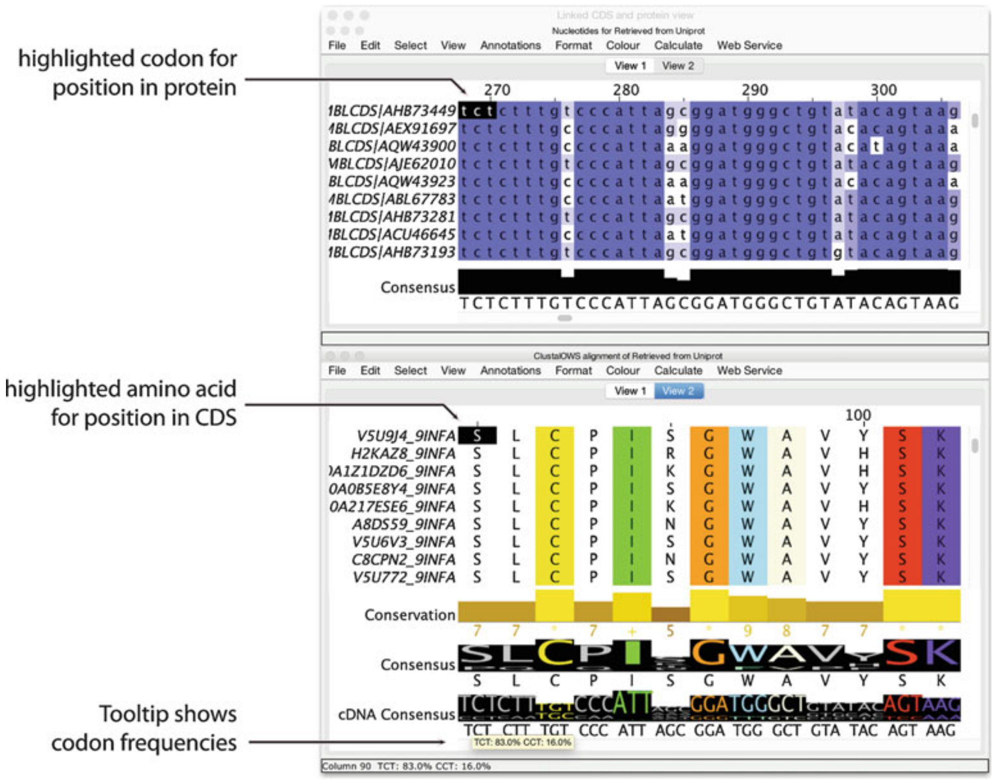

highlighted codon for position in protein
highlighted amino acid for position in CDS
Tooltip shows codon frequencies
Linked CDS and protein view
Nucleotides for Retrieved from Uniprot
File Edit Select View Annotations Format Colour Calculate Web Service
View 1 View 2
ClustalOWS alignment of Retrieved from Uniprot
Consensus
Conservation
cDNA Consensus
TCT: 83.0% CCT: 16.0%
Column 90 TCT: 83.0% CCT: 16.0%

Fig. 2 Jalview's linked CDS and Protein views. Screenshot of the Jalview 2.11 Desktop showing a reconstructed coding sequence alignment for a Clustal Omega alignment of influenza (H5N1) neuraminidase protein sequences. Protein sequences were retrieved from UniProt via Jalview's Sequence Fetcher and aligned with the Clustal Omega Web Service with default parameters. The coding sequence alignment was reconstructed by selecting "EMBLCDS" from the "Show Cross References" submenu of the "Calculate" menu, which triggered a retrieval of coding sequences from the European Nucleotide Archive. Two views have been created for the Linked CDS and Protein view (View 1 and View 2); the highlighted positions in CDS and Protein views are shown when the mouse is moved across the alignment area. Display of Sequence Logos have been enabled via the Consensus Annotation row's pop-up menu in order to display logos for amino acid and cDNA codon frequencies

5. When saving alignment data, annotations, and projects, Jalview may not always ask you to confirm if you are about to overwrite an existing file—particularly if you employ the "Save File" shortcut key (either CTRL-S or CMD-S on Macs). In these situations, Jalview will by default automatically create a backup file (called, e.g., myfile.fa.bak001). Backup behavior can be changed in the "Backups" section of Jalview's preferences panel: by default only the three most recent backups will be retained.
6. Alignment of large sets of sequences is CPU intensive and can therefore take considerable time. Jalview's public alignment services provided by the University of Dundee in Scotland,

UK, do not permit execution of alignments of greater than 1000 sequences with up to 1000 amino acids each. To perform larger alignments, it is necessary to either (a) download and configure a local instance of a compatible Jalview web services system or (b) align sequences using an external program and then import the result.

(a) *Downloading and configuring a local instance of Jalview Web Services*. Jalview versions prior to 2.12 are able to access web services provided by the JABAWS system (http://www.compbio.dundee.ac.uk/jabaws), and instructions for local installation either as a virtual appliance or tomcat web application are provided (http://www.compbio.dundee.ac.uk/jabaws/docs/getting_started.html). It may then be necessary to compile binaries for your platform and modify the JABAWS execution limits to permit alignments to be performed of the size that you require (http://www.compbio.dundee.ac.uk/jabaws/docs/advanced.html#limiting-the-size-of-the-job-accepted-by-jabaws). Once configured, Jalview can be connected to your new JABAWS server by entering its URL in Jalview's "Web Services" Preferences pane, and once validated, services will be accessible from the Web Services menu in the alignment view.

A new web services system is currently being developed, and instructions will be made available via Jalview's built-in help on how to download and deploy these new services once they are put into production.

(b) *Exporting sequences for alignment and reimporting the result*. Sequences may be exported via the File menu's "Save as" option in the alignment window or for the current selection via the pop-up menu (opened by right-clicking the selected area). Once the alignment has been performed, it is straightforward to import the aligned result to a new alignment view, but there are potential problems:

- Alignment programs may have constraints on sequence name length, the range of characters permitted, and reject inputs containing duplicate sequence names.
- Import of the aligned sequences as a new alignment will not retain sequence metadata from the original Jalview alignment view such as CDS relationships, database cross-references, sequence features, and secondary structure annotation rows.
- Hidden columns in the original view will not be accounted for or included in the result of the alignment (*see* **Note 9**).

Problems with sequence names can be worked around through the use of custom scripts, but are outside the scope of this chapter. Assuming sequence identifiers have been preserved, then there are two ways to work around the loss of sequence metadata. Metadata originally retrieved by Jalview from an external database can be retrieved once again. The "Fetch DB Refs" option in the Web services menu provides options to retrieve records from either all standard databases or a specific one (e.g. UniProt for protein sequences), but this should be used with care since both options may take some time for large alignments. A second workaround is to manually export features and annotation from the original view and import the resultant Jalview features file (or GFF3 file) and Jalview annotations file to the newly imported aligned sequence set.

7. Jalview provides a Realignment option when performing alignments via the Clustal W and Clustal Omega web service programs. Normally, Jalview removes all gap characters from sequences passed to an alignment program, but for Realignment, gap characters will be preserved. The precise behavior depends upon which Clustal alignment program is used:
   - Clustal W identifies aligned regions of the input as a range of sequences of equal length (including any gap characters) at the beginning of the input data. All other sequences in the input data are then aligned to that first block (with inserts into the block created as necessary). This process is quick, and preserves the original aligned region, provided that region was reordered to appear at the top of the alignment view when the alignment was submitted.
   - Clustal Omega realigns sequences by performing a sequence-profile alignment. A profile is first constructed from the input sequences (including any gap characters present). Gaps are then removed from all sequences, and they are each aligned to the profile to generate the final multiple sequence alignment result. This method is more computationally expensive than ClustalW, and it is unlikely that relationships between aligned sequences will be preserved in regions of poor alignment reliability in the result.
8. Many of Jalview's operations apply either to the whole alignment or, when present, just the selected region. Selected regions in Jalview are highlighted with a red box. Columns are annotated with a red mark, and selected sequences highlighted in dark gray. Rows and columns on alignment view can be selected by clicking and dragging with the mouse on the sequence ID panel and on the alignment ruler. An area of the

view can also be selected simply by clicking and dragging. These are summarized in the Jalview online video (http://www.jalview.org/videos/selectinginjalview) and FAQ "How Do I edit Sequences in Jalview" (http://www.jalview.org/faq#sequences). Functions in the Alignment window's select menu allow columns to be selected on the basis of alignment annotation (with the Select/Hide By Annotation dialog) and when regions of the alignment are highlighted as the result of a find operation. Columns can also be selected according to the presence of sequence features.

9. Selected regions in a view can be hidden or shown simply by pressing "H" or one of the other key combinations shown under the View menu's Hide and Show submenus. Columns containing gaps can also be hidden via the "Hide Inserts" function in the Selection and Sequence ID pop-up menu, opened by right-clicking in the sequence ID area. Hidden sequences and columns are excluded from alignment analysis (e.g., the conservation and consensus rows), tree calculation, principal component analysis, and secondary structure predictions (accessed via the submenu in the Web Services drop-down menu). Conversely, when multiple alignments are performed, hidden columns are "preserved" and not submitted to the server. Instead, the chosen program is executed several times, once for each contiguous region of the input set. Once all jobs are complete, Jalview concatenates the results and intervening hidden regions in order to construct the final alignment view.

10. Identification of reliably aligned regions in a multiple alignment is important for many applications. T-Coffee's Transitive Consistency Score (TCS) provides one approach: it measures the average shift error between the sets of positions aligned in a multiple alignment and a library of pairwise alignments involving the same sequences. Optimally aligned regions will always be aligned in the same way, and an increasing shift error is more indicative of low reliability. T-Coffee also offers a consensus alignment tool, M-COFFEE, which computes multiple alignments for the input sequences with several different multiple alignment programs and then generates a final alignment from these different results. Here, TCS scores reflect shifts between the different multiple alignment results used to generate the consensus.

11. A range of MSA column statistics are automatically computed and can be displayed as alignment annotation rows below the alignment. These rows update automatically as sequences are added or removed from the view, or the MSA is otherwise adjusted via Jalview's interactive MSA editing capabilities.

    The "Visual" tab in Jalview's user preferences dialog provides options for enabling or disabling Consensus, Occupancy, Conservation, and Alignment Quality Scores. The Consensus annotation row shows the modal residue in each column (or + if more than one residue is observed) and the proportion of sequences that contain that residue. Right-clicking on the Consensus row's annotation label (on the left-hand side) opens a pop-up menu that allows the consensus sequence for the view to be copied to the clipboard (and so pasted to a new alignment). Options in the menu also allow gapped sites to be ignored when computing the height of the consensus histogram and a Sequence Logo to be overlaid or shown in place of the histogram. Occupancy simply reflects the number of sequences that are aligned at each column of the MSA. Alignment Quality and Conservation are only available for Protein MSAs. The Alignment Quality score reflects the total likelihood of observing mutations between amino acids aligned at the given column, based on the BLOSUM62 [15] substitution matrix. The Conservation score for a column is computed according to Zvelebil et al. [16] as implemented in the AMAS method [17] and reflects the number of physicochemical properties shared by all amino acids in a column. The tooltip for each column lists conserved properties with properties prefixed with an exclamation mark (!) to indicate the absence of that property among the aligned residues. Jalview also provides access to AACon [18] through the Alignment Conservation submenu of the Web Services menu, which permits a further 17 conservation scores to be computed.

12. Jalview's built-in help provides a key and description for each of the color schemes available in the "Colour" menu. There are two classes of color scheme, symbol-based, such as Hydrophobic, or Taylor, and dynamic, such as ClustalX and Blosum62. When working with nucleotide alignments, the only dynamic scheme available is PID—which reflects abundance. The Purine and Pyrimidine color scheme, however, can be used to identify variation that may suggest differences in RNA secondary structure.

13. MSAs involving sequences that have diverse functions or complex evolutionary relationships such as duplications and domain expansions can be difficult to interpret for a number of reasons. The central problem is that while some columns are conserved, others are divergent as a result of evolutionary

pressure; and at each site, the degree of divergence between any two sequences may not always be the same as for others. For instance, orthologous sequences will exhibit conservation across functionally relevant regions, but those same functional regions may not be conserved in paralogs. As a consequence, global statistics such as consensus and conservation are not always sufficient to identify regions of alignments that are important for the structure and function of a sequence family. A fully automated method for the identification of such regions solely on the basis of sequence remains a research problem, but hierarchical alignment analysis methods such as AMAS [17] (and when a 3D structure is available, Evolutionary Trace [19]) can be effective. These approaches reveal local patterns of conservation and divergence by subdividing aligned sequences into clusters according to their percentage identity, ideally with a tree computed from the alignment (*see* **Note 14** below). Jalview enables alignments to be partitioned into groups in a similar manner, and its per-group conservation and consensus shading allow patterns of conservation to be revealed within each group. This approach does not on its own provide a way of quantifying the functional importance of a conserved region, but for proteins, Multi-Harmony [20] (via the Web Services' Analysis submenu) can be applied to an MSA with subgroups defined in order to infer columns that exhibit functional variation.

14. Jalview includes the algorithms "UPGMA" and Neighbor-Joining for the generation of dendrograms from distance matrices computed over a range of columns in an MSA. These functions are accessed from the "Calculate Tree or PCA..." dialog in the Calculations menu. A variety of score functions are provided [21] including protein substitution matrices such as BLOSUM62 and a Percent Identity score suitable for DNA. Trees calculated by external programs may also be imported as New Hampshire (Newick) and New Hampshire "Extended" format flat files—Jalview will attempt to automatically match leaves to sequences based on the displayed sequence IDs.
15. In addition to the Jalview Desktop application, Jalview is also available as a web-based application: JalviewJS. Launched in late 2019, JalviewJS is the Jalview application compiled to JavaScript [2] and adapted to run in-page either as the full-featured "Desktop" application or as interactive MSA visualization components designed for embedding in web pages. For more details, please *see* http://www.jalview.org/jalview-js/.

Acknowledgments

JBP and GJB wrote the manuscript. SLD created training videos based on exercises in the Jalview Manual developed by JBP and DMAM. All other authors contributed to Jalview development in the period 2009–2019. Development of the Jalview sequence analysis platform is coordinated at the University of Dundee with support from the UK's Biotechnology and Biological Sciences Research Council (BB/L010742/1), the Scottish Funding Council, and the Wellcome Trust (101651/Z/13/Z). We also gratefully acknowledge contributions and feedback from the Jalview user community and student project support from the Google Summer of Code (2011 and 2012).

References

1. Waterhouse AM, Procter JB, Martin DM, Clamp M, Barton GJ (2009) Jalview Version 2--a multiple sequence alignment editor and analysis workbench. Bioinformatics 25 (9):1189–1191. https://doi.org/10.1093/bioinformatics/btp033
2. Hanson R, Barton GJ, Procter J, Carstairs G, Soares B (2019) java2script/SwingJS for bioinformatics: reintroducing Jalview on the Web as JalviewJS. https://f1000research.com/posters/8-1578
3. The Jalview Web Site. (2019). http://www.jalview.org. Accessed 2nd Sept 2019
4. Troshin PV, Procter JB, Sherstnev A, Barton DL, Madeira F, Barton GJ (2018) JABAWS 2.2 distributed web services for Bioinformatics: protein disorder, conservation and RNA secondary structure. Bioinformatics 34 (11):1939–1940. https://doi.org/10.1093/bioinformatics/bty045
5. Troshin PV, Procter JB, Barton GJ (2011) Java bioinformatics analysis web services for multiple sequence alignment—JABAWS:MSA. Bioinformatics 27(14):2001–2002. https://doi.org/10.1093/bioinformatics/btr304
6. Moretti S (2019) Jalview 2.11.0 package for Homebrew https://formulae.brew.sh/cask/jalview
7. Gruning B, Dale R, Sjodin A, Chapman BA, Rowe J, Tomkins-Tinch CH, Valieris R, Koster J, Bioconda T (2018) Bioconda: sustainable and comprehensive software distribution for the life sciences. Nat Methods 15 (7):475–476. https://doi.org/10.1038/s41592-018-0046-7
8. Madeira F, Madhusoodanan N, Lee J, Tivey ARN, Lopez R (2019) Using EMBL-EBI Services via web interface and programmatically via web services. Curr Protoc Bioinformatics 66(1):e74. https://doi.org/10.1002/cpbi.74
9. Yates A, Beal K, Keenan S, McLaren W, Pignatelli M, Ritchie GR, Ruffier M, Taylor K, Vullo A, Flicek P (2015) The Ensembl REST API: Ensembl data for any language. Bioinformatics 31(1):143–145. https://doi.org/10.1093/bioinformatics/btu613
10. Wallace IM, O'Sullivan O, Higgins DG, Notredame C (2006) M-Coffee: combining multiple sequence alignment methods with T-Coffee. Nucleic Acids Res 34 (6):1692–1699. https://doi.org/10.1093/nar/gkl091
11. Chang JM, Di Tommaso P, Notredame C (2014) TCS: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. Mol Biol Evol 31(6):1625–1637. https://doi.org/10.1093/molbev/msu117
12. Schneider TD, Stephens RM (1990) Sequence logos: a new way to display consensus sequences. Nucleic Acids Res 18 (20):6097–6100. https://doi.org/10.1093/nar/18.20.6097
13. Yachdav G, Wilzbach S, Rauscher B, Sheridan R, Sillitoe I, Procter J, Lewis SE, Rost B, Goldberg T (2016) MSAViewer: interactive JavaScript visualization of multiple sequence alignments. Bioinformatics 32 (22):3501–3503. https://doi.org/10.1093/bioinformatics/btw474
14. Ofoegbu TP, James B. Procter (2015) BioJSON Version 1.0 Schema for representation and exchange of annotated Multiple Sequence Alignments with The Jalview Workbench.

https://jalview.github.io/biojson/v1.0/. Accessed 2nd Sept 2019

15. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A 89(22):10915–10919. https://doi.org/10.1073/pnas.89.22.10915
16. Zvelebil MJ, Barton GJ, Taylor WR, Sternberg MJ (1987) Prediction of protein secondary structure and active sites using the alignment of homologous sequences. J Mol Biol 195 (4):957–961. https://doi.org/10.1016/0022-2836(87)90501-8
17. Livingstone CD, Barton GJ (1993) Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. Comput Appl Biosci 9(6):745–756
18. Golicz AT, Troshin PV, Madeira F, Martin DMA, Procter JB, Barton GJ (2018) AACon: A Fast Amino Acid Conservation Calculation Service. http://www.compbio.dundee.ac.uk/aacon/. Accessed 2nd Sept 2019
19. Lichtarge O, Bourne HR, Cohen FE (1996) An evolutionary trace method defines binding surfaces common to protein families. J Mol Biol 257(2):342–358. https://doi.org/10.1006/jmbi.1996.0167
20. Brandt BW, Feenstra KA, Heringa J (2010) Multi-harmony: detecting functional specificity from sequence alignment. Nucleic Acids Res 38(Web Server issue):W35-W40. doi:https://doi.org/10.1093/nar/gkq415
21. The Calculate Tree or PCA Dialog. (2019). http://www.jalview.org/help/html/calculations/tree.html. Accessed 2nd Sept 2019
22. Sali A, Blundell TL (1993) Comparative protein modelling by satisfaction of spatial restraints. J Mol Biol 234(3):779–815. https://doi.org/10.1006/jmbi.1993.1626
23. Webb Bea (2010) Alignment File (PIR)—from the Modeller Manual Online. https://salilab.org/modeller/9v8/manual/node454.html. Accessed 2nd Sept 2019
24. Sussman JL, Lin D, Jiang J, Manning NO, Prilusky J, Ritter O, Abola EE (1998) Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. Acta Crystallogr D Biol Crystallogr 54(Pt 6 Pt 1):1078–1084. https://doi.org/10.1107/s0907444998009378
25. Westbrook JD, Bourne PE (2000) STAR/mmCIF: an ontology for macromolecular structure. Bioinformatics 16(2):159–168. https://doi.org/10.1093/bioinformatics/16.2.159
26. Felsenstein J (1986) The Newick Tree Format. http://evolution.genetics.washington.edu/phylip/newicktree.html. Accessed 2nd Sept 2019
27. Cardona G, Rossello F, Valiente G (2008) Extended Newick: it is time for a standard representation of phylogenetic networks. BMC Bioinformatics 9:532. https://doi.org/10.1186/1471-2105-9-532

Chapter 14

Evolutionary Sequence Analysis and Visualization with Wasabi

Andres Veidenberg and Ari Löytynoja

Abstract

Wasabi is an open-source, web-based graphical environment for evolutionary sequence analysis and visualization, designed to work with multiple sequence alignments within their phylogenetic context. Its interactive user interface provides convenient access to external data sources and computational tools and is easily extendable with custom tools and pipelines using a plugin system. Wasabi stores intermediate editing and analysis steps as workflow histories and provides direct-access web links to datasets, allowing for reproducible, collaborative research, and easy dissemination of the results. In addition to shared analyses and installation-free usage, the web-based design allows Wasabi to be run as a cross-platform, stand-alone application and makes its integration to other web services straightforward.

This chapter gives a detailed description and guidelines for the use of Wasabi's analysis environment. Example use cases will give step-by-step instructions for practical application of the public Wasabi, from quick data visualization to branched analysis pipelines and publishing of results. We end with a brief discussion of advanced usage of Wasabi, including command-line communication, interface extension, offline usage, and integration to local and public web services. The public Wasabi application, its source code, documentation, and other materials are available at http://wasabiapp.org

Key words Evolutionary sequence analysis, Reproducible research, Data visualization, Web application

1 Introduction

In evolutionary sequence analysis, phylogenetic trees and multiple sequence alignments are tightly linked. Many analyses use one in the inference of the other, e.g., a guidetree to infer an alignment, or an alignment to infer a phylogenetic tree. Some sequence aligners (e.g., PRANK [1] and PAGAN [2]) and many downstream evolutionary analysis tools and pipelines (e.g., CodeML [3], EPO [4]) combine phylogenetic and sequence data to infer parameters attached to specific nodes of input trees. Some parameters relate only to the tree while others, like ancestral sequences, are associated both on the tree and the input alignment. To get the necessary context for drawing conclusions, such parameters should be displayed together with both input datasets.

Wasabi [5] was designed to work with complex phylogenetic datasets, displaying each sequence next to the corresponding tree node and maintaining the link through tree edits and downstream analysis steps. In addition to data visualization, Wasabi integrates external programs, editing tools, data management, and related functions into a user-friendly graphical interface, providing a comprehensive environment for phylogenetic sequence analysis. While there are other software packages with a more versatile tools selection (e.g., Mega [6], ETE [7]), Wasabi is characterized by its web-based implementation. Use of modern web technologies allows Wasabi to provide, among other features, installation-free access, fine-grained customizations, secure linking to datasets, and a plugin system for extending its functionality.

The first two sections below provide an overview of Wasabi and an example workflow of a Wasabi analysis. They describe Wasabi in a public web service configuration (as used in http://wasabiapp.org), where a central analyses database is accessed via user accounts and sharing URLs. After that, we briefly discuss alternative setups of Wasabi, its modifications with plugins, and other advanced topics. Throughout the chapter, *italics* is used to mark the terms found in the Wasabi interface, and underline for typed text, filenames, and web addresses. Consecutive actions (typically mouse clicks) are linked with arrows (→). Some paragraphs are supplemented with a note section to add details to the main text.

2 Overview of the User Interface

Wasabi's graphical user interface is arranged to a horizontal toolbar placed on top of the visualization area that include sections for rendering phylogeny, taxa names, and multiple sequence alignment (*see* Fig. 1). This layout is adjustable: the top toolbar can be contracted or collapsed to maximize the visualization space (use *Tools* → *Settings*) and the vertical divider lines between the visualization sections are draggable to change their relative width. Most of the interface elements (e.g., buttons, icons, or text with dotted underline) reveal a tooltip describing the associated function when pointed with a mouse cursor for a few seconds. Colored text indicates links that open sections (blue links) or windows (red links) within the Wasabi interface.

The top toolbar includes buttons for drop-down menus, visualization zoom level, undo/redo, and notifications. Most of the tasks in Wasabi are done via dialog windows listed in the toolbar menus. To reduce interface clutter, only the applicable tasks are visible. For example, the *Data* menu initially shows just the *Import*

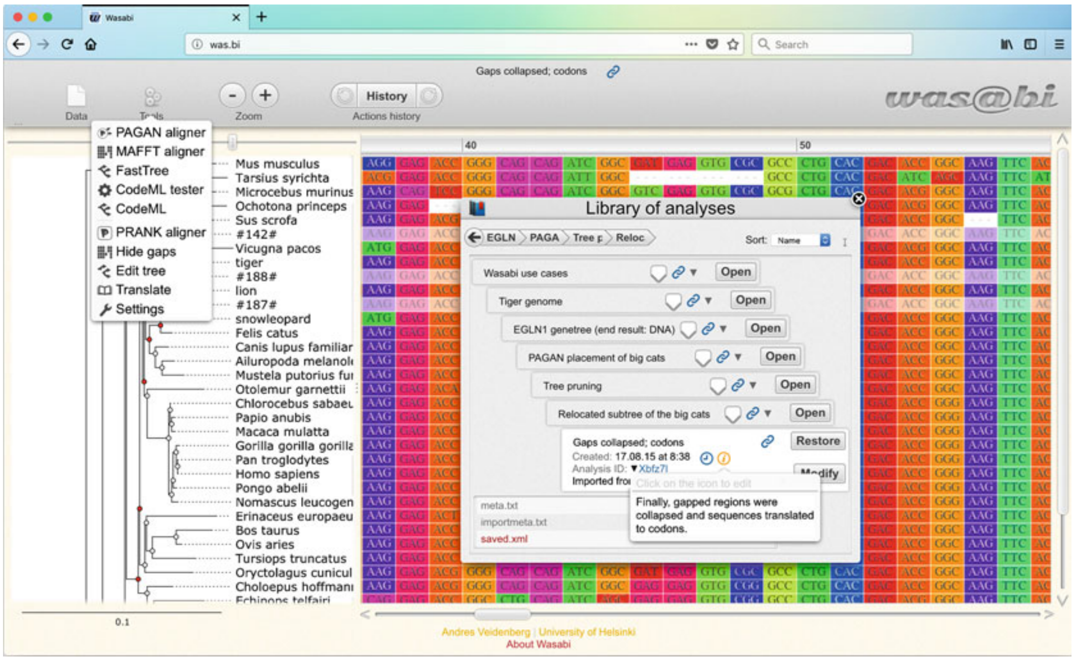


Fig. 1 Wasabi with an imported analysis library dataset. Here, some ancestral sequences have been revealed, the *Tools* menu is open, and the *Analysis library* shows the imported analysis step together with its workflow path, annotation text and the file list

tool. After a dataset is imported, the menu is expanded with *Export* and *Info* tools. Logging to a user account (*see* the example workflow below) adds *Analysis library*, *Save*, and *Share* options.

The *Import* tool accepts tree and/or sequence data input from local or remote sources. It auto-detects common file standards (FASTA, ClustalW [8], Phylip [9], Newick [10], NHX [11], NEXUS [12], HSAML [13], PhyloXML [14]) but can also handle unknown data formats. The input sequence type (DNA/RNA/protein) is set automatically but, if needed, it can be manually corrected in the *Info* tool. The *Import* window includes a file drop area and selector button for local files, a dedicated section for importing phylogenomic datasets from the Ensembl database [15] and a multisource text field accepting data files from a web address, Wasabi dataset ID, or raw text data. The right side of the text field is accompanied by a clickable triangle to expand the text field and a "plus"-marked button for importing multiple files. Once a dataset has been imported, it can be converted to another format in the *Export* tool (supports FASTA, Phylip, Newick, NHX, HSAML, and NEXUS).

The *Tools* menu lists the integrated command-line analysis programs, built-in data editing tools, and system settings. The selection of available tools depends on the installed plugins and the type of input data imported to Wasabi. At the time of writing and using the default plugins, the following tools appear when the currently open dataset includes:

- Sequences: PRANK [1], PAGAN [2] and MAFFT [16] sequence aligners, FastTree [17] tree inference method, and *Hide gaps* tool.
- A tree: *Edit tree* tool.
- Both sequences and a tree: CodeML [3] selection model tester.

Adding custom programs to Wasabi is described in the plugins section. The built-in *Hide gaps* tool allows masking, collapsing, or removing gap-rich or conserved sequence alignment columns. *Edit tree* tool is useful for fine-grained tree modifications, including collapsing/removing specific taxa, adding annotations (e.g., branch colors) or preparing the tree for running CodeML branch-site models [18]. The *Settings* window contains (depending on the Wasabi configuration) up to 30 adjustable preferences, including autosave, color schemes for visualization, and the user account management. Many of the options are concealed in collapsed sections that, like elsewhere in Wasabi interface, are marked with triangle-shaped text bullets. The hidden content can be revealed (or rehidden) with a mouse click on the bulleted text line.

Each time the user saves an imported dataset (*Data* → *Save*) or runs an analysis program (*Tools* menu), a data snapshot is stored to the user account. Together with other snapshots this forms a workflow track in the *Analysis library.* If the input dataset was imported from an external source, the snapshot is stored as the first step of a new workflow; otherwise the save location can be set either to continue or to branch off from the input analysis step. The stored analysis histories are listed in the *Analysis library* window. The analysis step currently open is marked with a white background while read-only shared analyses have dashed borders (*see* **step 8** in the tutorial). A click on the arrowhead button of any analysis step reveals the subsequent step on the analysis path, while a click on the breadcrumb path bar or the back button takes a step toward the root. The *Ladderized* and *Compact* layout modes in the library window, found under the gear button or *Tools* → *Settings*, are useful when the list of stored analyses grows. Each analysis step includes info fields that can be revealed by clicking the black triangle. Some info fields can only be read by hovering its icon or title text (e.g., the date stamp from the clock icon, or the launch parameters from the program name), while others can be clicked and edited (e.g., the descriptive name) or open further options (a click on the dataset

ID allows accessing the stored files). One can remove an analysis step with its *Modify* button or import the default output file by clicking *Open* (one can switch the default file by opening it in the file list). In addition, the info icon allows displaying and editing a free-text annotation and the link icon shows the dedicated sharing link. While Wasabi automatically creates a workflow of subsequent analysis steps, root level analysis steps can be collected into larger analysis collections by dragging them by their left side and dropping onto other analyses.

When the imported dataset includes both a phylogenetic tree and sequences with matching taxon names, sequences in the alignment area are displayed next to their position in the phylogenetic tree. Ancestral sequences are hidden by default but can be revealed via drop-down menu by clicking any tree node. The menu also gives access to the node metadata and allows modifying the connected subtree (show/hide/remove/recraft/reroot). Ancestral nodes can also be dragged to relocate them or to remove specific clades. Individual annotation labels and coloring displayed on the tree can be defined in the *Settings* window or modified with the *Edit tree* tool. Specific sequence alignment columns can be masked, collapsed or removed by dragging a selection box spanning the chosen sites, followed by the selected task in the right-click menu. Collapsed sequence sites are indicated with red markers on the ruler bar running along the top edge of the alignment box. The ruler also serves as a dragging handle for panning around the alignment (as an alternative to using arrow keys, mouse scroll wheel, or the scrollbars).

In addition to the visualization and built-in tools, the user interface wraps complex functionality like the analysis database and background processes that are perhaps best described with a practical analysis workflow.

3 Example Workflow

3.1 Introduction

This tutorial is significantly updated and expanded version of the workflow published in the original Wasabi article [5] that verified the findings of a study [19] linking snow leopard's high-altitude adaptation to amino acid changes in the hypoxia-related gene EGLN1. In short, multiple sequence alignment of EGLN1 is created by merging homologous sequences from Ensembl database with the study data, then realigned with two alternative methods, cleaned, and finally tested for signals of positive selection. Every analysis step with intermediate results is automatically visualized, stored to analysis history, and accessible through the web via sharing URLs.

Although the tutorial includes a detailed list of steps to cover most of the tools and functionality available in Wasabi, in practice the workflow is fairly simple and straightforward: the introduction video featured on the Wasabi homepage summarizes most of the process in under 2 minutes (*see* Fig. 2 for overview). Also, you can pick and choose individual tutorial steps to form shorter tasks. For example, Wasabi is often used for quickly visualizing a sequence alignment file by dropping it to the importer (*see* **step 1**), as an interactive tree editor (**step 5**), versatile file converter (**step 2**), or to make a dataset available across the web (**step 8**).

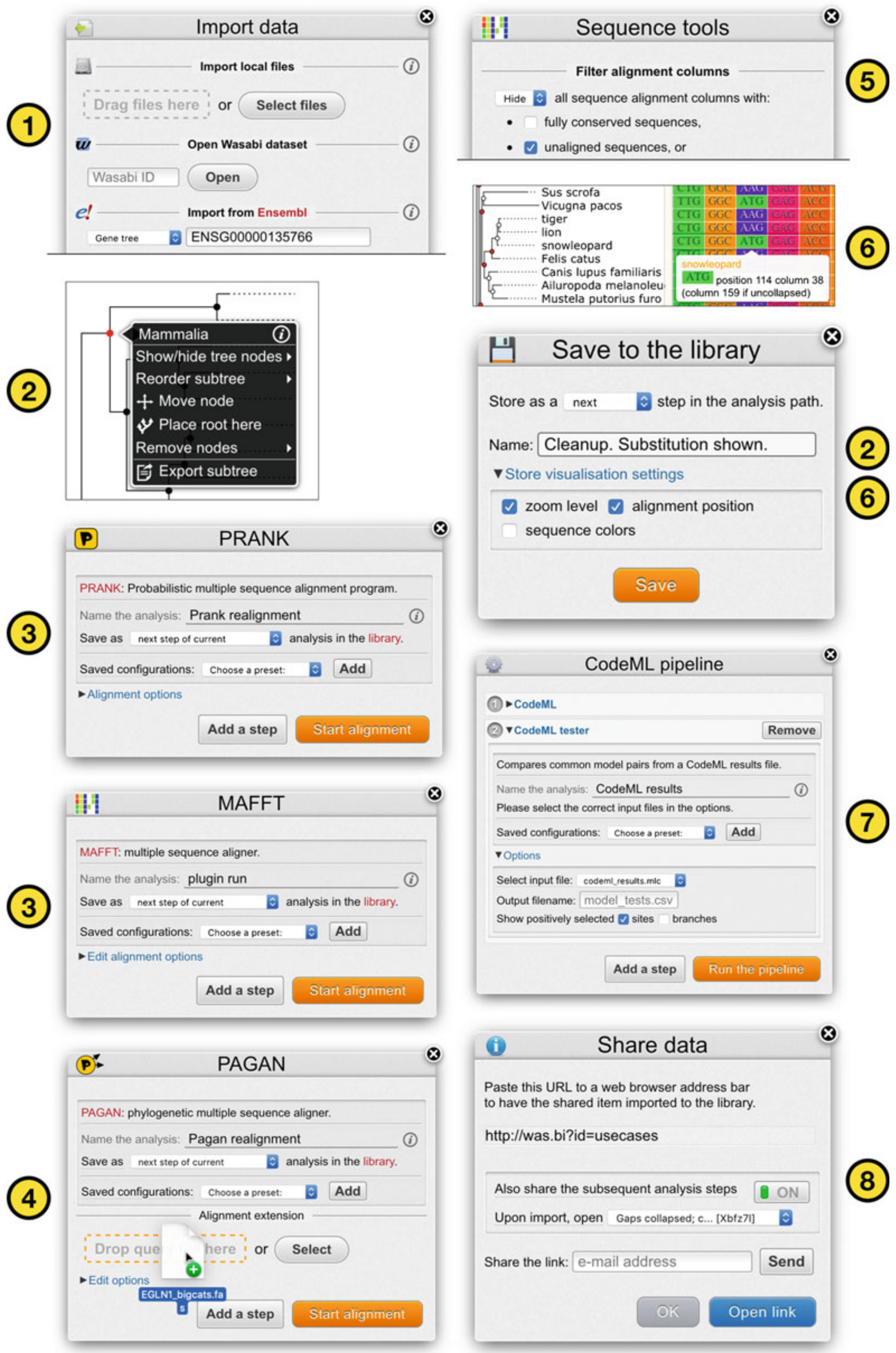
3.2 Setup

Open the Wasabi application by clicking the launch button on http://wasabiapp.org (or go directly to http://was.bi). When visiting Wasabi for the first time (or using the web browser in incognito mode), the *Create account* notification will show up on the top toolbar. Wasabi user accounts allocate a 100 MB server space for storing datasets and running background jobs. You can dismiss the notification when using Wasabi for just visualizing, editing, and exporting datasets. For enabling full functionality (used in the tutorial from **step 2** onward), click the notification button and fill in your email address. Wasabi sends a message to this address when the account is created, about to expire (after 30 days of the last visit), or when a background job has finished (optional). Alternatively, you can opt for a temporary account (valid for 1 day) without entering an email address.

Note that, after clicking *Create account*, Wasabi's web address has changed (to the form was.bi/yourUserID). This address is a direct link to your Wasabi user account and allows you to open your analysis library on any internet-connected device. Please keep the address for future reference, as otherwise you will lose the access to your stored datasets after the Wasabi window has been closed. Here are some suggestions for storing and retrieving your account URL:

- Write the address down or bookmark it in the web browser.
- Enable "Remember me on this computer" in the confirmation window. The web browser will automatically redirect to the account address on subsequent Wasabi launches.
- Locate the link in the email message that Wasabi sent you when the account was created.

In addition to the user account, the tutorial assumes that you have a query file ready to be used in **step 4**. Download it from http://wasabiapp.org/download/wasabi/other/EGLN1_bigcats.fas.



PRANK: Probabilistic multiple sequence alignment program.
Name the analysis: Prank realignment
Save as next step of current analysis in the library.
Saved configurations: Choose a preset: Add
Alignment options

MAFFT: multiple sequence aligner.
Name the analysis: plugin run
Save as next step of current analysis in the library.
Saved configurations: Choose a preset: Add
Edit alignment options

PAGAN: phylogenetic multiple sequence aligner.
Name the analysis: Pagan realignment
Save as next step of current analysis in the library.
Saved configurations: Choose a preset: Add

Hide all sequence alignment columns with:
- ☐ fully conserved sequences,
- ✓ unaligned sequences, or

position 114 column 38 (column 159 if uncollapsed)

Store as a next step in the analysis path.
Name: Cleanup. Substitution shown.
✓ zoom level ✓ alignment position
☐ sequence colors

Compares common model pairs from a CodeML results file.
Name the analysis: CodeML results
Please select the correct input files in the options.
Saved configurations: Choose a preset: Add
Select input file: codeml_results.mlc
Output filename: model_tests.csv
Show positively selected ✓ sites ☐ branches

Paste this URL to a web browser address bar to have the shared item imported to the library.
http://was.bi?id=usecases
Also share the subsequent analysis steps ON
Upon import, open Gaps collapsed; c... [Xbfz7l]
Share the link: e-mail address

Fig. 2 Overview of the example workflow, represented by Wasabi interface cutouts. Circled numbers indicate corresponding tutorial steps

3.3 Instructions

Step 1: Import EGLN1 Gene Sequences: First, open the import tool (*Data → Import*) to download a GeneTree [20] dataset with EGLN1 homologs (*see* **Note 1**). In the Ensembl section, choose "Gene tree" from the left-hand selection, click *Import options*, type human and EGLN1 to the *species* and *gene name* fields, choose cDNA, and click *Get ID*. When the GeneTree ID appears to the top input field, click *Import*. After a brief moment, the tree and sequences are rendered to the visualization area and the import window disappears.

Step 2: Reduce the Dataset: In this step, the set of included EGLN1 sequences is reduced to mammalian species. Locate the most recent common ancestor of the mammals clade by hovering the mouse cursor over the tree inner nodes until the label displays *Mammals* (*see* **Note 2**). Click the node to reveal a pop-up menu. Select *Remove nodes → Keep only subtree*.

3.3.1 Optional: Store a Data Snapshot

At any point during the tutorial, the currently open dataset can be browsed, modified, downloaded in a desired file format (*Data → Export*), or stored to the *Analysis Library* (*Data → Save*). Although optional (the next step stores the current data as input), a snapshot of the current dataset is handy for a couple of reasons: it serves as the root step for the following analysis pipeline, and gets a dedicated URL for sharing it in the web and to other Wasabi accounts (*see* **step 8**). When a background job is run (tutorial **steps 3, 4**, and 7), an analysis snapshot (including input, output, and metadata) is automatically added to the analysis history in the library. Whenever the dataset state is not stored to the database, it's indicated on the toolbar *(unsaved)*. The undo button allows reversing data edits up to the last snapshot.

Step 3: Realign with PRANK: Since the end goal of the workflow is to study positive selection, it's recommended to realign the mammalian EGLN1 sequences with an aligner designed for evolutionary analyses. Click *Tools → PRANK aligner* and type a descriptive name for this analysis step (e.g., "Prank realignment"). Next, open the *Alignment options → Fine tuning* section and tick *align as codons*. Click *Start alignment*. Click the notification button on the toolbar to check the status of the running background jobs. When the PRANK alignment has finished, click *Open* to import the results.

3.3.2 Optional: Realign with MAFFT

The EGLN1 sequences could also be aligned with another aligner to create an alternative starting point for the rest of the workflow (*see* **Note 3**). This would be useful to, e.g., estimate the sensitivity of the positive selection tests to the choice of the alignment method. Start the MAFFT realignment (*Tools→MAFFT aligner*) with default settings and let it run in the background. This will create two independent realignments (PRANK and MAFFT version) from the same input dataset, creating a branching point in the analysis path. Go straight to **step 4** to complete the rest of the tutorial, then return here to continue with the alternative analysis branch. Wasabi will take care of recording both workflows.

Unlike PRANK, MAFFT does not output a guide tree with taxa names matching the output alignment. Since the next tutorial step needs a reference phylogeny, a new tree needs to be built for the MAFFT alignment. Make a new tree with *Tools→FastTree*, open the resulting dataset and continue with the next tutorial step.

Step 4: Add More Sequences: Next, the EGLN1 alignment is extended with homologs from the species studied in the snow leopard paper (tiger, lion, and snow leopard). Click *Tools→PAGAN aligner* and drag the EGLN1_bigcats.fas file (from the setup step) to the query file drop area. Edit the name field for a better description. This setup will use PAGAN to extend the currently open alignment with the sequences from the query file. After clicking *Start alignment*, the notification button and *Status overview* window will show the progress of the PAGAN execution (*see* **Note 4**).

Step 5: Cleanup: After the PAGAN alignment has finished, open the results and then browse the imported alignment. Remove low-quality sequences (showing long stretches of missing data) and paralogs (species duplicates) by dragging the taxa name out of the tree and release it to the trash bin-marked alignment area (or click taxa name → *Remove leaf*). The placement of the added sequences depends on the reference alignment quality and sequence similarity. Check the location of the big cats and recraft if needed: drag and drop to the leopard/lion ancestral node to the cat branch. Next, trim out gappy alignment columns (uninformative for the following selection tests): click *Tools* → *Hide gaps*, adjust the sequence rows threshold to a low value (e.g., 4%) and click *Apply.* Then, right-click the alignment area → *Remove hidden columns* to delete the collapsed gap sites. If you rearranged the tree, the alignment does not match it anymore and needs to be updated. Click the *Realign* notification on the toolbar, tick *use codon model*, and click *Update alignment.* After the realignment, the imported dataset will update itself and a data snapshot is added to the analysis library workflow path.

Step 6: Browse: Click *Tools* → *Translate* and select *codons.* Browse the sequence alignment and locate the AAG → ATG substitution (Lys → Met, at around site 40 on the alignment ruler) in the snow leopard (*see* **Note 5**). EGLN1 has been annotated as a hypoxia-related gene and the source study linked this amino acid change to snow leopard's adaptation to high-altitude environment. The conclusion is supported by the identical substitution at the same alignment position in the alpaca. (Alpaca was not part of the original study but was included here from the GeneTree.) A quantitative confidence score can be added to this finding with a positive selection test.

Step 7: Test for Positive Selection: The site-wise positive selection test implemented in CodeML calculates (among other metrics) an estimate of selection (d_N/d_S ratio) and its confidence score (p-value) for each site in the input multiple sequence alignment. The results file, however, is excessively detailed and makes eyeballing for relevant info a time-consuming task (especially when processing multiple genes). In addition, the validity of the column scores needs to be confirmed by comparing the overall fit of the positive selection model to an alternative model. Therefore, Wasabi's toolset includes a script that parses CodeML result files, performs likelihood ratio tests [21] for the compared model pairs and extracts statistically significant column scores. The script (like the rest of integrated programs) can be chained to a pipeline to feed the results from one analysis step (model testing) to the next one (results parsing).

Click *Tools → CodeML*. Type a new analysis step name. Select *Multiple site ratios*. Click *Edit options*, select *single ratio* for the branch model (*see* **Note 6**). Expand *Site model* and tick models 1 (nearly neutral) and 2 (positive selection). Now, without closing the CodeML window, open *Tools → CodeML tester* (or *Add a step → CodeML tester*). Note that the CodeML section is collapsed and the parser program is added as a second step, forming a pipeline. Optionally, tick *send an email when the pipeline finishes*. This setup will run CodeML to test the two selected models against the input alignment data and to calculate the site scores, followed by the CodeML results parser. Click *Run pipeline*.

Step 8: Review and Publish: Follow the pipeline progress in the *Status overview* window and click the *Open* button once it appears. This time, the results are not in the imported tree and alignment (that originate from **step 5**), but in the CodeML parser report file. To access it, open *Analysis library* and locate the active analysis step (marked with white background and labeled with the name from **step** 7). Click the analysis ID to see the list of files, hover model_tests.csv, and click the revealed *View* button. The file content is opened in a separate window. If everything went as planned, the report is expected to indicate that the data support the presence of positive selection (model M2 passed the hypothesis test) at the previously noted substitution site (dn/ds ratio > 1 with p-value $<5\%$).

At this point, you have verified the Lys → Met change in EGLN1 gene from the snow leopard study and improved the confidence of the finding by including more sequence data and running a statistical test. You have also collected a detailed record of the analysis process. You can go back and examine each step in the *Analysis library*, visualize the intermediate results, check the analysis program parameters and input files, or split the analysis path to alternative branches (*see* **step 3** for an example branching point).

To share your findings with the academic community, click on the blue link icon on any workflow step in the library. The resulting *Share data* window gives you the option to either share only the output dataset of the selected step, or together with the subsequent workflow. You can then distribute the displayed sharing URL, e.g., via email or social media (*see* **Note** 7). The link will launch Wasabi on any internet-connected device, open the shared dataset and (if enabled) will add the included workflow as a read-only copy to the recipient's *Analysis library.* This allows the recipient to view and work with the received datasets without affecting the source, store the modifications and send back the updated version via another sharing URL.

In addition to online communication, the sharing links can be used in scientific publishing. For example, the Wasabi article [5] includes an image of the EGLN1 alignment with the snow-leopard specific substitution, accompanied by a sharing URL (http://was.bi?id=usecases). When the reader clicks the link, Wasabi is launched, visualizing the EGLN1 alignment at the same position as depicted in the figure. The reader can then browse the rest of the alignment and all of the steps in the analysis workflow. A neat attribute of the shared workflow is that when we modify it (perhaps to fix an error), the distributed copies are also automatically updated in the recipient libraries. Similar sharing link, together with representative dataset and annotations, can also be created for our tutorial workflow. Start with an empty analysis collection (*Analysis library* → gear button → *New collection*). Drag the workflow (e.g., its root step) into the new collection. Click the collection's sharing icon and set the data snapshot from **step 6** as the default dataset in the "Upon import..." selection menu. Update the annotation (click the collection's info icon) with a free-text description and links to reference articles [19]. Your tutorial workflow is now ready and wrapped and should look quite similar to our version: http://was.bi?id=tutorial.

4 Advanced Topics

4.1 Under the Hood

Wasabi is built upon modern web technologies, consisting of the main application (written in JavaScript), a server component (written in Python), and third-party programs (plugins). The modular design allows for cross-platform support (including mobile devices), different setup configurations, and extensibility. For example, Wasabi can be launched on a local computer as a desktop application, run on a server computer to provide a web service, or integrated into an existing web page.

Wasabi is installed by downloading its files from http://wasabiapp.org/downloads. The application can be used without the server module by opening the index.html file: this allows for import, visualization, editing, and export of datasets. The full functionality with external tools and the analysis library is enabled by launching the server script (wasabi_server.py). Wasabi is then available from the server's local address (by default http://localhost:8000). When Wasabi is running on a computer reachable from the internet, it can be provided as a web service, allowing for quick access, data sharing and central updates. Wasabi's server module uses job scheduling and user accounts for managing system resources and randomized IDs for data security. Wasabi running modes, user account quotas and other application parameters can be edited in the server module settings file (wasabi_settings.cfg).

The analysis library uses a file-based database, where the folder structure represents analysis paths with metadata stored in meta.txt files. This allows direct reading or writing to the analysis library in a file browser or on the command line. An existing data folder can be defined as an analysis library in the settings file. The Wasabi interface only shows folders with a meta.txt file (that, at the minimum, should include the ID and name fields).

The Wasabi application communicates with its server component and the outside world with URL commands and be used to integrate Wasabi with other tools and websites. For example, a command-line pipeline can use wget [22] to retrieve or write files to a remote Wasabi analysis library, or open a web browser to visualize a dataset with locally installed Wasabi. Also, Wasabi links are an easy way to add a visualizer to a web-based alignment database. The URL parameters in the links can be used to provide a customized visualizer, e.g., to disable specific functions, display bootstrap values, or hide the toolbar. *See* http://wasabiapp.org/rest for documentation and examples.

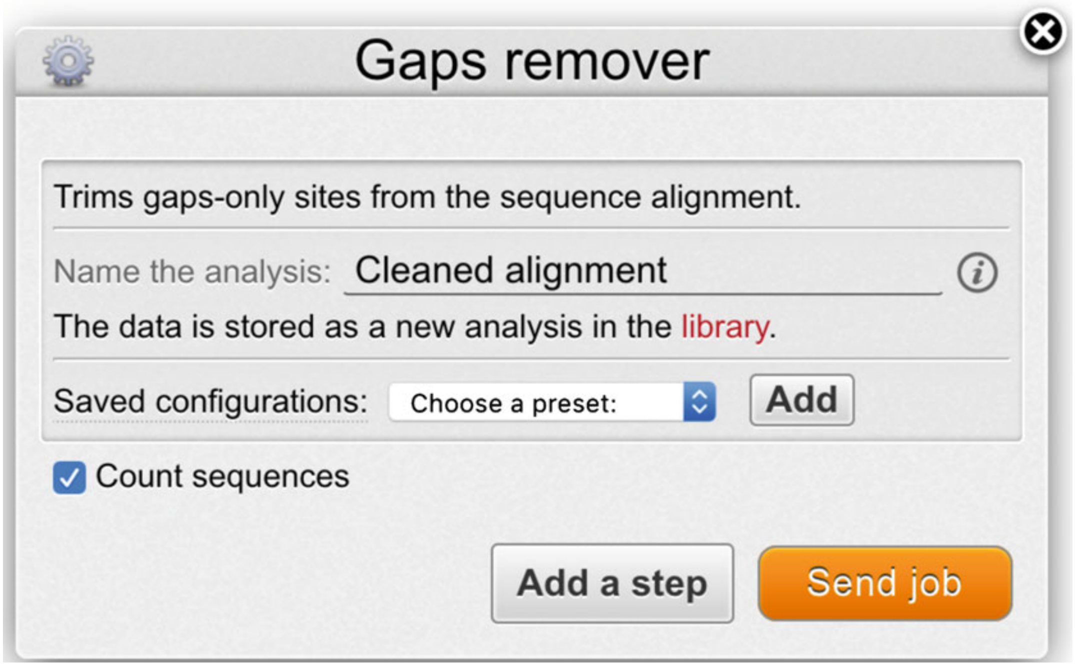
Since Wasabi is a web application, it can be added to an existing web service like any other web page: by including Wasabi's HTML, CSS and JavaScript files, and linking to the Wasabi's URL where needed. In addition, Wasabi's appearance and functionality can be extensively customized by editing the style.css and script.js files. Examples of web services with integrated and customized Wasabi include the Silva rRNA database (http://www.arb-silva.de), the ConSurf conservation profile database (http://consurfdb.tau.ac.il), and the Ensembl genome browser (http://ensembl.org).

In comparison to a native compiled application, a downside of a web-based implementation is the performance cost arising from the web browser overhead. Wasabi uses several optimization strategies to scale well even with large input datasets. For example, the sequence data is rendered to static but graphics card accelerated canvas elements that are expanded piece-by-piece as new alignment regions are scrolled into the viewport. Since the imported dataset is loaded to memory, the maximum dataset size is limited mainly by the RAM available in the computer. We have tested Wasabi on a regular laptop with up to 1-gigabyte data files, showing performance on par with native visualization programs [5]. A favorable side effect of the web browser environment is that, with the continuous development of JavaScript engines, the performance of Wasabi improves over time even without contributions from the future code optimizations.

4.2 Plugins

Wasabi utilizes a plugin system to integrate external tools to its graphical interface. A plugin is a JSON-formatted [23] description of a command-line program that Wasabi uses to communicate with the program and to construct a graphical user interface for launching the tool. Figure 3 depicts a JSON specification for a python script with two input parameters. The script is integrated to Wasabi by dropping it together with the JSON file to the plugins folder (*see*

```
{
    "program": "remove_gaps.py",   //executable
    "name": "Gaps remover",
    "desc": "Trims gaps-only sites from the sequence alignment",
    "outfile": "output.fa",
    "options": [  //input parameters
        {"file": "imported sequences"},
        {"checkbox": "Count sequences", "option": "--count"}
    ]
}
```



Gaps remover

Trims gaps-only sites from the sequence alignment.

Name the analysis: Cleaned alignment

The data is stored as a new analysis in the library.

Saved configurations: Choose a preset: Add

☑ Count sequences

Add a step Send job

Fig. 3 A minimal example of a plugin JSON file (top) and the resulting Wasabi interface window (bottom)

Note 8). The command-line program now appears in the *Tools* menu and has gained a graphical interface. The interface also allows chaining the plugins to form analysis pipeline that can be stored together with the filled parameters for later reuse. As demonstrated by the CodeML interface, the plugin API [24] allows building complex interfaces with embedded instructions, alternative parameter sets, user input-dependent default values, etc. The JSON files for current Wasabi tools are found in the plugins folder and the full API documentation is available at http://wasabiapp.org/plugins.

5 Future Directions

Wasabi has grown from its beginnings as a capable cross-platform sequence alignment visualizer to an extendable analysis platform for evolutionary sequence analyses, with a distinctive user-friendly interface and easy access across the web. While there is no shortage of directions for improvement, some features planned for the upcoming Wasabi updates include a substring search, image file export, reference sequence support, and a visualization track for plotting site-wise graphs or annotations. Other improvements under consideration aim to reduce Wasabi's memory footprint and simplify integration to web content.

Wasabi's plugin system allows using standardized program descriptions for adding command-line tools to the graphical interface, significantly reducing the workload and know-how required for implementing the integration. Although the plugin system was built for Wasabi, it could be useful for many other programs and provide them with a dynamic graphical interface. To help in that, Wasabi's plugin system is now provided as a separate JavaScript library, the universal interface generator Pline. With that, one can quickly build graphical interfaces to command-line tools and use them on any web page or as a stand-alone desktop application. Each plugin file should only be written once, and then updated together with the target program. To that end, an online repository for sharing and reusing JSON files is available on the Wasabi homepage.

6 Notes

1. This procedure imports a dataset from Ensembl database. The same input dataset could have been fetched in the import tool via other routes, e.g., by dragging a prepared data file (perhaps from a previous run of this tutorial) to the file drop area, typing http://rest.ensembl.org/genetree/member/id/ENSG00000135766 to the bottom section input field (this

address uses Ensembl REST API [25]), or by filling the same input with the dataset ID g7TDxl to import a copy from our Wasabi account.

2. Dragging the tree/alignment divider to the right and using the vertical scrollbar will help to browse big trees.
3. An analysis path can be split at any time from a selected step in the *Analysis library*. For example, if you stored the input dataset in **step 2** (you can also do it now since the dataset is still open), you can skip the branching option for now and return to this step later. For that, open the data snapshot (from **step 2**) in the library window and continue the tutorial with the MAFFT alignment step. Likewise, you don't have to finish the tutorial in one go. You can close and later relaunch Wasabi, open a stored analysis step and resume from there, or even switch between the alternative analysis paths while completing the remaining steps.
4. You can terminate a running background process with the *Kill* button. A terminated (or failed) process will show a *Delete* button: clicking that removes the generated files. You can also skip the *Open* button and move the files directly to the *Analysis library* for inspection (click the gear icon → *Move to library*). Also, hovering the *program name* will show its launch parameters and clicking the blue *feedback* text line reveals the full output log.
5. After you have located the substitution site, it's a good idea to save a data snapshot with the *zoom level* and *alignment position* enabled in the *Store visualization* settings section. Next time you (or a colleague using the sharing URL from **step 8**) opens the stored dataset, the alignment will automatically move to the correct position without having to spend time to relocate the substitution.
6. Hover the mouse over an input field to see the associated command-line parameter. The adjacent info icon or dotted text reveals the relevant documentation.
7. If you are using Wasabi datasets in a scientific publication, make sure that the sharing links will stay permanently accessible. The public Wasabi (http://wasabiapp.org) is an academically funded, free-of-charge service. At the time of writing, we provide free user accounts and keep user data for a minimum of 30 days but we cannot guarantee long-term storage of external datasets. However, you can easily install Wasabi locally (in a server configuration) to share your datasets permanently.
8. It's recommended to add a copy of the target program to the plugins folder instead of using a system-wide command, since future program updates may break the plugin compatibility.

References

1. Löytynoja A (2014) Phylogeny-aware alignment with PRANK. Methods Mol Biol 1079:155–170
2. Löytynoja A, Vilella AJ, Goldman N (2012) Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. Bioinformatics 28:1684–1691
3. Yang Z (2007) PAML 4: phylogenetic analysis by maximum likelihood. Mol Biol Evol 24:1586–1591
4. Paten B, Herrero J, Beal K et al (2008) Enredo and Pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. Genome Res 18:1814–1828
5. Veidenberg A, Medlar A, Löytynoja A (2016) Wasabi: an integrated platform for evolutionary sequence analysis and data visualization. Mol Biol Evol 33:1126–1130
6. Kumar S, Stecher G, Li M et al (2018) MEGA X: molecular evolutionary genetics analysis across computing platforms. Mol Biol Evol 35:1547–1549
7. Huerta-Cepas J, Dopazo J, Gabaldón T (2010) ETE: a python environment for tree exploration. BMC Bioinformatics 11:24
8. Larkin MA, Blackshields G, Brown NP et al (2007) Clustal W and Clustal X version 2.0. Bioinformatics 23:2947–2948
9. Baum BR (1989) PHYLIP: Phylogeny Inference Package. Version 3.2. Joel Felsenstein. Q Rev Biol 64:539–541
10. Felsenstein J (2004) Inferring Phylogenies. Sinauer Associates Incorporated
11. Zmasek CM NHX—New Hampshire eXtended, version 2.0. http://phylosoft.org/NHX/. Accessed 19 Aug 2019
12. Maddison DR, Swofford DL, Maddison WP (1997) NEXUS: an extensible file format for systematic information. Syst Biol 46:590–621
13. Löytynoja A. HSAML format. http://wasabiapp.org/software/hsaml_format/. Accessed 19 Aug 2019
14. Han MV, Zmasek CM (2009) phyloXML: XML for evolutionary biology and comparative genomics. BMC Bioinformatics 10:356
15. Zerbino DR, Achuthan P, Akanni W et al (2018) Ensembl 2018. Nucleic Acids Res 46: D754–D761
16. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30:772–780
17. Price MN, Dehal PS, Arkin AP (2010) FastTree 2--approximately maximum-likelihood trees for large alignments. PLoS One 5:e9490
18. Zhang J, Nielsen R, Yang Z (2005) Evaluation of an improved branch-site likelihood method for detecting positive selection at the molecular level. Mol Biol Evol 22:2472–2479
19. Cho YS, Hu L, Hou H et al (2013) The tiger genome and comparative analysis with lion and snow leopard genomes. Nat Commun 4:2433
20. Vilella AJ, Severin J, Ureta-Vidal A et al (2009) EnsemblCompara GeneTrees: complete, duplication-aware phylogenetic trees in vertebrates. Genome Res 19:327–335
21. Yang Z, Nielsen R, Goldman N, Pedersen AM (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. Genetics 155:431–449
22. GNU Wget. https://www.gnu.org/software/wget/wget.html. Accessed 19 Aug 2019
23. The JSON Data Interchange Syntax. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. Accessed 19 Aug 2019
24. Wasabi plugin API. http://wasabiapp.org/about/wasabi-plugin-api/. Accessed 19 Aug 2019
25. Yates A, Beal K, Keenan S et al (2015) The Ensembl REST API: Ensembl data for any language. Bioinformatics 31:143–145

Chapter 15

Seaview Version 5: A Multiplatform Software for Multiple Sequence Alignment, Molecular Phylogenetic Analyses, and Tree Reconciliation

Manolo Gouy, Eric Tannier, Nicolas Comte, and David P. Parsons

Abstract

We present Seaview version 5, a multiplatform program to perform multiple alignment and phylogenetic tree building from molecular sequence data. Seaview provides network access to sequence databases, alignment with arbitrary algorithm, parsimony, distance and maximum likelihood tree building with PhyML, and display, printing, and copy-to-clipboard or to SVG files of rooted or unrooted, binary or multifurcating phylogenetic trees. While Seaview is primarily a program providing a graphical user interface to guide the user into performing desired analyses, Seaview possesses also a command-line mode adequate for user-provided scripts. Seaview version 5 introduces the ability to reconcile a gene tree with a reference species tree and use this reconciliation to root and rearrange the gene tree. Seaview is freely available at http://doua.prabi.fr/software/seaview.

Key words Multiple sequence alignment, Molecular phylogeny, Phylogenetic tree building, Graphical user interface, Tree reconciliation

1 Seaview and Its Context

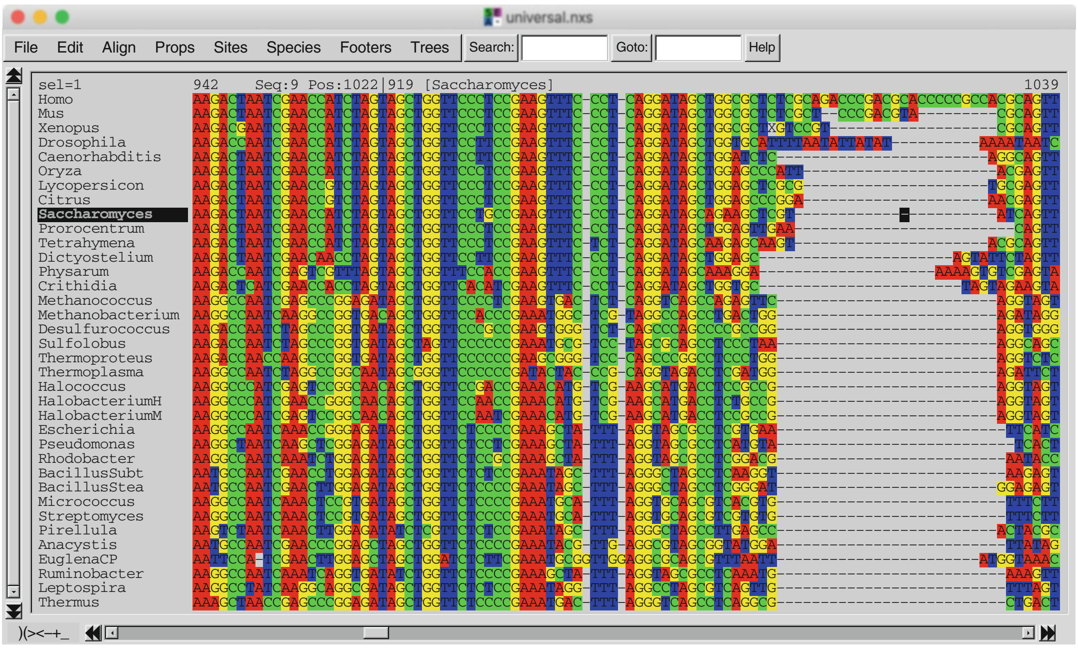
Many molecular evolution analyses depend on two key tasks to be performed in succession, multiple sequence alignment, and phylogenetic tree reconstruction. Both tasks represent vast, mature research areas which have led to the development of a large number of algorithms, each with one or several implementations [1]. Seaview [2] was designed to facilitate performing these tasks by providing a graphical user interface and by taking care of all the data handling steps necessary before, between and after these tasks. Furthermore, Seaview is a multiplatform program that can run on all computers in wide use today. Overall, a nonspecialist user of Seaview can read a dataset of DNA or protein sequences, extend it with homologous sequences from the international nucleotide sequence database collaboration [3], align them, estimate the phylogenetic tree that describes the evolutionary history of the divergence of these sequences, print the resulting tree, and paste it to a drawing software to prepare it for publication, all this without leaving the Seaview program. Additionally, and this is new in Seaview version 5, the program performs gene tree/species tree reconciliations, that is, annotating duplications, speciations, and losses in a gene tree, and can automatically rearrange branches with low statistical support when this can remove dubious duplications. Graphical visualization of gene trees inside a species tree is included.

Seaview is organized around three major windows: the alignment window, the tree window, and the help window. The latest version of Seaview, adequate to most currently used computers, can be downloaded visiting its dedicated web page [4]. Seaview is also available as a package for several major Linux distributions, including Debian which currently provides Seaview version 4.7 [5].

2 The Alignment Window

2.1 Visualization of Multiple Sequence Alignments

The alignment window (Fig. 1) provides a view into the set of DNA or protein sequences under analysis: it shows *w* residues (nucleotides or amino acids) of *h* sequences alongside the sequence names. The number of residues and sequences displayed is determined, respectively, by the width and height of the window. A horizontal and a vertical slider are allowed to scroll through the sequences.



```
sel=1          942    Seq:9 Pos:1022|919 [Saccharomyces]          1039
Homo
Mus
Xenopus
Drosophila
Caenorhabditis
Oryza
Lycopersicon
Citrus
Saccharomyces
Prorocentrum
Tetrahymena
Dictyostelium
Physarum
Crithidia
Methanococcus
Methanobacterium
Desulfurococcus
Sulfolobus
Thermoproteus
Thermoplasma
Halococcus
HalobacteriumH
HalobacteriumM
Escherichia
Pseudomonas
Rhodobacter
BacillusSubt
BacillusStea
Micrococcus
Streptomyces
Pirellula
Anacystis
EuglenaCP
Ruminobacter
Leptospira
Thermus
```

Fig. 1 Example of an alignment window

There is no a priori limit to the number and length of sequences Seaview can handle, nor to the length of sequence names. They are limited by the memory capacity of the host computer. Thus, a Seaview user running an average PC can visualize alignments of several thousand sequences without difficulty. Seaview is practical to handle sequence alignments of the scale of one to a few tens of genes but will not be convenient to work at the scale of full eukaryotic chromosomes. As shown below, Seaview uses external programs to perform multiple alignment which determine the scale of the number of sequences that can be aligned in reasonable time (*see* Chapter 1 for Clustal Ω). Tree building by distance methods are very rapid, so that phylogenetic trees for several hundreds of sequences can be easily computed. Tree building by the maximum likelihood approach is much more computer intensive, so that Seaview is convenient for datasets containing less than hundred sequences. Larger problems are more conveniently handled by running program PhyML in command-line mode, and having Seaview graphically display the resulting tree at the end of the computation. Several alignment windows can be opened simultaneously, and sequences can be copy/pasted between them. Seaview handles six formats able to contain sequence data: Fasta, MSF, Mase, Phylip, Clustal, and NEXUS. MSF and Mase have a historical origin and are no longer in common use. Fasta is useful because it's a sequence file format that many programs read and write and that can be easily produced manually. Formats Phylip and Clustal allow to read the output of two widely used pieces of software: PHYLIP [6] for phylogenetic analysis and Clustal Ω for multiple sequence alignment [7]. The NEXUS format [8] originates from the PAUP software [9] and is used now by many other bioinformatics programs. It is especially useful because it can store all the data handled by Seaview in a single file: sequences, sequence descriptions, sequence subsets, site subsets, and trees. The "File" menu allows to open a data file and to save all or part of the window content to the same or another file in any of the six known formats.

2.2 Selecting Sequences and Sites

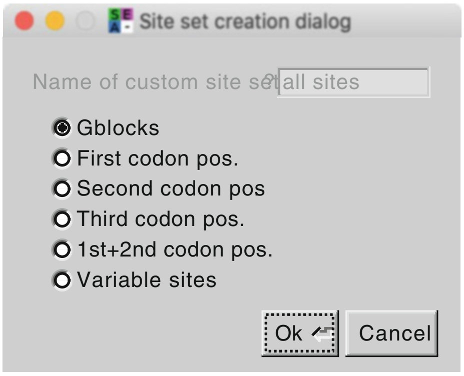
Seaview allows the user to select sequences and/or sites to restrict further operations to the selected sites/sequences alone.

Sequences are selected by clicking on their names (click and drag to select several sequences at once). A selected subset of sequences can be memorized and given a name using the "Species" menu, they will then be saved in the data file.

Sites of interest upon the sequences can be defined using the "Sites" menu. These sites can be defined by various criteria (Fig. 2).

Site selection criteria are

– Selected sites can be manually defined by clicking and dragging on a dedicated pseudo sequence appearing at the bottom of the alignment window.



Name of custom site set: all sites

[✓ Gblocks]
First codon pos.
Second codon pos
Third codon pos.
1st+2nd codon pos.
Variable sites

Fig. 2 Site set creation dialog window

- Evolutionarily conserved sites can be identified using the Gblocks algorithm [10].
- Seaview can select all first, second, third, or first + second codon positions of the current dataset.
- Seaview can select variable sites, or, in other words, omit conserved sites.

Any operation (e.g., alignment and tree building) performed while a selection of sequences and/or sites is active will be restricted to the selection rather than being performed on the whole dataset.

2.3 The "Align" Menu

The "Align" menu of Seaview alignment windows contains all alignment-related features: the user chooses what alignment algorithm to use, sets options for that algorithm, and runs it on all sequences or on the selected part of the selected sequences. The "Alignment options" submenu lists the alignment algorithms Seaview can perform and allows to set options for them. It also allows to add external alignment methods to Seaview (Fig. 3).

Seaview is distributed bundled with two multiple sequence alignment methods: Clustal Ω [7] (*see* Chapter 1) and muscle [11]. Both can be used to align either nucleotide or protein sequences. Besides these two alignment methods, Seaview contains a mechanism that allows to have it pilot other sequence alignment programs. The only requirements for a program to be added to Seaview's list of alignment methods are that it can read a Fasta-formatted input file, it can write the resulting aligned sequences to a Fasta-formatted file, and it can be run by a command line containing the names of those two Fasta files. The "Add external method" menu item visible in Fig. 3 adds one multiple sequence alignment algorithm to the list of algorithms Seaview can pilot. When this operation is performed, two pieces of data are required: (1) the path to the desired alignment program on the local computer; (2) the series of arguments this program expects to read a Fasta file, align its

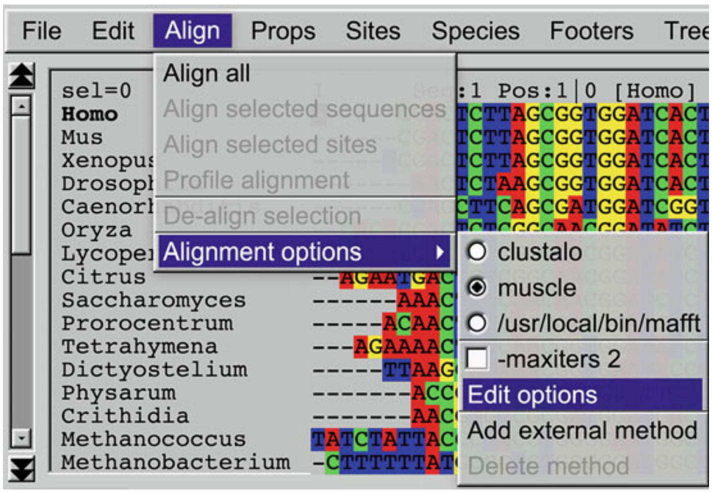


Fig. 3 The “Align” menu and its “Alignment options” submenu. “-maxiters 2” is an option to the MUSCLE method, which can be turned on or off using the associated check box. “Edit options” allows to set or change what options appear in the menu. The “MAFFT” algorithm has been added as an extra alignment tool for Seaview to pilot

content, and output the alignment to a Fasta file. In these arguments, the input file is to be denoted by %f.pir and the output file by %f.out. For the MAFFT algorithm [12], for example, these arguments are.

```
“--auto %f.pir > %f.out”.
```

For the T_Coffee method [13] (*see* Chapter 6), these arguments are.

```
“%f.pir -outfile=%f.out -output=fasta_aln”.
```

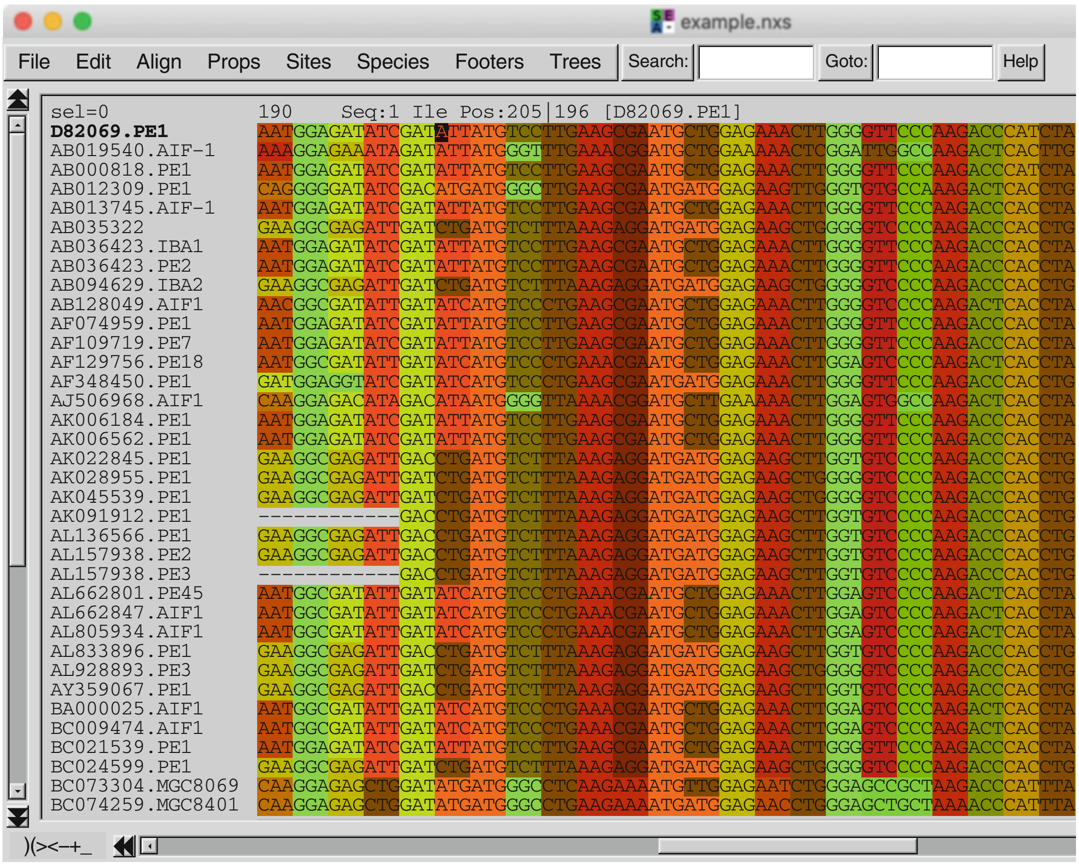
For the Probcons program [14], the argument string becomes.

```
“%f.pir > %f.out”.
```

The “Add external method” procedure is to be done once by the user for Seaview to be able to use the added method in all further runs.

2.4 Protein-Coding DNA Sequences

Protein-coding DNA sequences need to be aligned taking their codon structure into account to obtain a correct alignment. Seaview performs this through the “View as proteins” item of the “Props” menu of the alignment window. When the window contains protein-coding sequences and “View as proteins” is turned on, DNA sequences are replaced by their corresponding protein



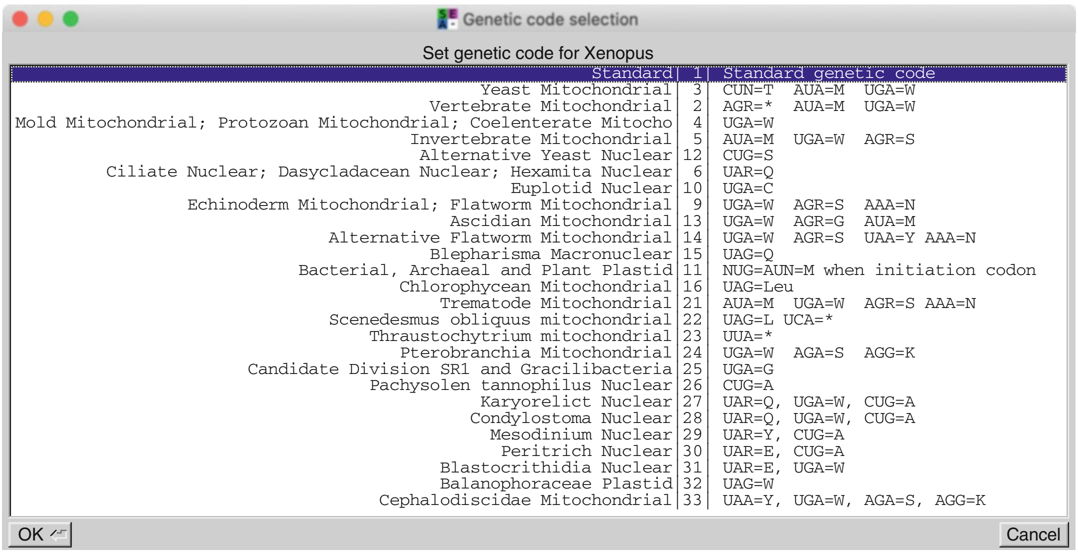
```
sel=0              190     Seq:1 Ile Pos:205|196 [D82069.PE1]
D82069.PE1         AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCATCTA
AB019540.AIF-1     AAAGGAGAAATAGATATTATGGGTTTGAAACGGATGCTGGAAAAACTCGGATTGGCCAAGACTCACTTG
AB000818.PE1       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCATCTA
AB012309.PE1       CAGGGGGATATCGACATGATGGGCTTGAAGCGAATGATGGAGAAGTTGGGTGTGCCAAAGACTCACCTG
AB013745.AIF-1     AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AB035322           GAAGGCGAGATTGATCTGATGTCTTTAAAGAGGATGATGGAGAAGCTGGGGGTCCCCAAGACCCACCTG
AB036423.IBA1      AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AB036423.PE2       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AB094629.IBA2      GAAGGCGAGATTGATCTGATGTCTTTAAAGAGGATGATGGAGAAGCTGGGGGTCCCCAAGACCCACCTG
AB128049.AIF1      AACGGCGATATTGATATCATGTCCCTGAAGCGAATGCTGGAGAAACTTGGGGTCCCCAAGACTCACCTA
AF074959.PE1       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AF109719.PE7       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AF129756.PE18      AATGGCGATATTGATATCATGTCCCTGAAACGAATGCTGGAGAAACTTGGAGTCCCCAAGACTCACCTA
AF348450.PE1       GATGGAGGTATCGATATCATGTCCCTGAAGCGAATGATGGAGAAACTTGGGGTTCCCAAGACCCACCTG
AJ506968.AIF1      CAAGGAGACATAGACATAATGGGGTTAAAACGGATGCTTGAAAAACTTGGAGTGGCCAAGACTCACCTA
AK006184.PE1       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AK006562.PE1       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
AK022845.PE1       GAAGGCGAGATTGACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
AK028955.PE1       GAAGGCGAGATTGATCTGATGTCTTTAAAGAGGATGATGGAGAAGCTGGGGGTCCCCAAGACCCACCTG
AK045539.PE1       GAAGGCGAGATTGATCTGATGTCTTTAAAGAGGATGATGGAGAAGCTGGGGGTCCCCAAGACCCACCTG
AK091912.PE1       ------------GACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
AL136566.PE1       GAAGGCGAGATTGACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
AL157938.PE2       GAAGGCGAGATTGACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
AL157938.PE3       ------------GACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
AL662801.PE45      AATGGCGATATTGATATCATGTCCCTGAAACGAATGCTGGAGAAACTTGGAGTCCCCAAGACTCACCTA
AL662847.AIF1      AATGGCGATATTGATATCATGTCCCTGAAACGAATGCTGGAGAAACTTGGAGTCCCCAAGACTCACCTA
AL805934.AIF1      AATGGCGATATTGATATCATGTCCCTGAAACGAATGCTGGAGAAACTTGGAGTCCCCAAGACTCACCTA
AL833896.PE1       GAAGGCGAGATTGACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
AL928893.PE3       GAAGGCGAGATTGATCTGATGTCTTTAAAGAGGATGATGGAGAAGCTGGGGGTCCCCAAGACCCACCTG
AY359067.PE1       GAAGGCGAGATTGACCTGATGTCTTTAAAGAGGATGATGGAGAAGCTTGGTGTCCCCAAGACCCACCTG
BA000025.AIF1      AATGGCGATATTGATATCATGTCCCTGAAACGAATGCTGGAGAAACTTGGAGTCCCCAAGACTCACCTA
BC009474.AIF1      AATGGCGATATTGATATCATGTCCCTGAAACGAATGCTGGAGAAACTTGGAGTCCCCAAGACTCACCTA
BC021539.PE1       AATGGAGATATCGATATTATGTCCTTGAAGCGAATGCTGGAGAAACTTGGGGTTCCCAAGACCCACCTA
BC024599.PE1       GAAGGCGAGATTGATCTGATGTCTTTAAAGAGGATGATGGAGAAGCTGGGGGTCCCCAAGACCCACCTG
BC073304.MGC8069   CAAGGAGAGCTGGATATGATGGGCCTCAAGAAAATGTTGGAGAATCTGGGAGCCGCTAAGACCCATTTA
BC074259.MGC8401   CAAGGAGAGCTGGATATGATGGGCCTGAAGAAAATGATGGAGAACCTGGGAGCTGCTAAAACCCATTTA
```

Fig. 4 The "Codon colors" display mode of protein-coding DNA sequences. The column containing the black cursor displays with the same red color synonymous Isoleucine codons (ATC, ATT, ATA) and uses other colors for non-synonymous ones (e.g., ATG for Methionine)

sequences in the window. These can be aligned with the "Align" menu. Finally, "View as proteins" can be turned off. DNA sequences reappear in the window aligned in such a way that each length-1 gap in the protein-level alignment is replaced by a length-3 gap at the DNA level.

Seaview proposes an alternative way to display protein-coding DNA sequences that highlights their codon structure. It's obtained with item "Colors/Codon colors" of the "Props" menu and displays synonymous codons of the same amino acid with the same color (Fig. 4).

Seaview can display, for instance using its "Codon colors" mode, multiple sequence alignments output by program MACSE [15] (*see* Chapter 4) able to align frameshift-containing sequences because of pseudogenes or sequencing errors: the extra "!" characters added by MACSE are accepted by Seaview.



Set genetic code for Xenopus

Standard	1	Standard genetic code
Yeast Mitochondrial	3	CUN=T AUA=M UGA=W
Vertebrate Mitochondrial	2	AGR=* AUA=M UGA=W
Mold Mitochondrial; Protozoan Mitochondrial; Coelenterate Mitocho	4	UGA=W
Invertebrate Mitochondrial	5	AUA=M UGA=W AGR=S
Alternative Yeast Nuclear	12	CUG=S
Ciliate Nuclear; Dasycladacean Nuclear; Hexamita Nuclear	6	UAR=Q
Euplotid Nuclear	10	UGA=C
Echinoderm Mitochondrial; Flatworm Mitochondrial	9	UGA=W AGR=S AAA=N
Ascidian Mitochondrial	13	UGA=W AGR=G AUA=M
Alternative Flatworm Mitochondrial	14	UGA=W AGR=S UAA=Y AAA=N
Blepharisma Macronuclear	15	UAG=Q
Bacterial, Archaeal and Plant Plastid	11	NUG=AUN=M when initiation codon
Chlorophycean Mitochondrial	16	UAG=Leu
Trematode Mitochondrial	21	AUA=M UGA=W AGR=S AAA=N
Scenedesmus obliquus mitochondrial	22	UAG=L UCA=*
Thraustochytrium mitochondrial	23	UUA=*
Pterobranchia Mitochondrial	24	UGA=W AGA=S AGG=K
Candidate Division SR1 and Gracilibacteria	25	UGA=G
Pachysolen tannophilus Nuclear	26	CUG=A
Karyorelict Nuclear	27	UAR=Q, UGA=W, CUG=A
Condylostoma Nuclear	28	UAR=Q, UGA=W, CUG=A
Mesodinium Nuclear	29	UAR=Y, CUG=A
Peritrich Nuclear	30	UAR=E, CUG=A
Blastocrithidia Nuclear	31	UAR=E, UGA=W
Balanophoraceae Plastid	32	UAG=W
Cephalodiscidae Mitochondrial	33	UAA=Y, UGA=W, AGA=S, AGG=K

Fig. 5 List of variant genetic codes. The first item of the list is the standard, universal genetic code. Each variant genetic code has a name (at left) as given by INSDC. The right part of each item details how the variant code differs from the standard code listing all sets of codons which are translated differently (* indicates stop codons)

2.5 Genetic Code Variants

Seaview can associate the adequate genetic code variant with the sequences it processes: item "Set genetic code" of the "Edit" menu. The user selects the relevant genetic code among all known variants (Fig. 5). Seaview uses all genetic codes defined by the International Nucleotide Sequence Database Collaboration (INSDC) [3] which are primarily based on two review articles [16, 17]. All protein-translation operations performed by Seaview use the genetic code associated to each sequence.

2.6 Closely Related Sequences

When dealing with closely related sequences, it may be difficult to visualize rare differences within long stretches of identical nucleotides. The item "by Reference" of the "Props" menu, active when exactly one sequence is selected, modifies the sequence display to help detect sequence variants: the selected sequence is moved to the top and other sequences display a residue where they differ from that at the same position in the reference sequence and a dot where it's identical.

2.7 The "Trees" Menu of Alignment Windows

Phylogenetic tree building is performed under Seaview via items of the "Trees" menu of an alignment window (Fig. 6).

The first three items of the "Trees" menu give access to three families of tree-building methods: parsimony, methods based on pairwise phylogenetic distances, and PhyML, a maximum likelihood-based method. The tree may be built from a subset of the data using site and/or sequence selections. At the end of a tree-

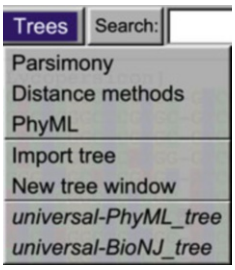


Fig. 6 The “Trees” menu of an alignment window, expanded with two previously computed trees

building operation, Seaview displays a new window containing the resulting phylogenetic tree (*see* Subheading 3, below).

The next item of the “Trees” menu allows to import an external tree in the form of a Newick-formatted [18] file. Such imported tree can be used, for example, to evaluate its likelihood and compare it to that of the most likely one. The “New tree window” item opens an empty tree window where it’s possible to paste a Newick-formatted tree from the computer’s clipboard.

The rest of the “Trees” menu gets dynamically populated with italicized names of trees handled by Seaview: both trees that were computed by Seaview and imported trees can be stored in that menu and later saved together with the aligned sequence data, provided the NEXUS or Mase formats are used. When Seaview reads in that saved file later, the trees it contains reappear at the end of the “Trees” menu. Selection of one of those menu items draws the corresponding tree in a tree window.

2.8 Parsimony-Based Tree Building

Seaview implements tree building by parsimony using, with authorization from the author, the dnapars and protpars programs from the PHYLIP package [6] which deal with DNA and protein sequences, respectively. The dnapars algorithm applied to DNA sequences produces a tree with branch lengths. These lengths are estimated according to the method of Hochbaum and Pathria [19] which “averages the number of reconstructed changes of state over all sites over all possible most parsimonious placements of the changes of state among branches” [6]. The protpars algorithm produces a tree without branch lengths displayed by Seaview such that all root-to-leave distances are equal. In both cases, the resulting tree is unrooted in essence. Seaview displays these trees in a rooted form; the Subheading 3 below describes how unrooted trees are drawn in a rooted form.

Figure 7 shows the dialog presented to the user to define the parsimony analysis to be performed on DNA sequences. The two boxes at the top, labeled “Randomize seq. order” and “times,” control whether the heuristic search within the space of alternative

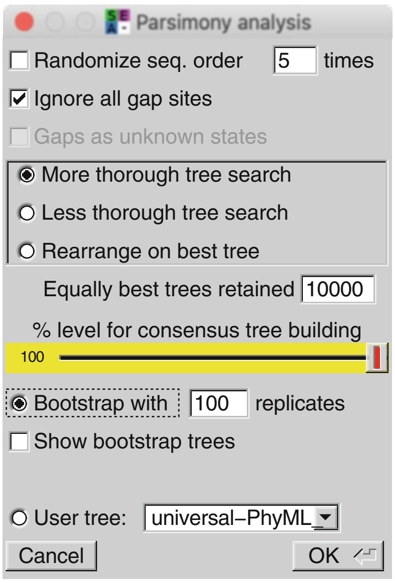

Parsimony analysis
Randomize seq. order 5 times
[✓ Ignore all gap sites]
Gaps as unknown states
[✓ More thorough tree search]
Less thorough tree search
Rearrange on best tree
Equally best trees retained 10000
% level for consensus tree building
100
[✓ Bootstrap with] 100 replicates
Show bootstrap trees
User tree: universal–PhyML
Cancel
OK

Fig. 7 The Parsimony dialog window

tree topologies is repeated several times after randomly changing the input order of data sequences.

The boxes labeled "Ignore all gap sites" and "Gaps as unknown states" control how gap-containing sites will be handled. When the first box is checked, any gap-containing site is completely ignored. Consequently, the parsimony operation only handles nucleotide substitutions or amino acid replacements. The difficulty to weigh the cost of an insertion/deletion event relatively to that of a substitution vanishes. Any phylogenetic information associated to shared insertions or deletions between sequences is ignored. When both boxes are unchecked, gaps are treated as an extra character state. Consequently, shared insertions or deletions between sequences bring information taken into consideration by the parsimony algorithm. But an insertion or deletion event of length n is given the cost of n independent substitution events, which may not be biologically realistic. When only "Gaps as unknown states" is checked, sequence gaps are treated as absence of sequence data. Consider a sequence site containing a gap in some sequences and regular nucleotides or amino acids for other sequences. That site does contribute to the parsimony score when gap-free sequences are compared, but the same site does not contribute to the parsimony score if at least one gap-containing sequence is involved.

The next three radio buttons control the search space considered while looking for the most parsimonious tree. More precisely, they define how to consider equally parsimonious candidate trees and branches of these trees without evidence that there is any change on them: the default option labeled "More thorough tree search" is to be preferred. However, a large dataset may render this option computationally intractable. One may then use one of the other two options, which increasingly reduce the computation time by producing less thorough searches of the space of candidate trees. The protpars algorithm used with protein datasets doesn't offer a comparable option.

"% level for consensus tree building" labels a slider set by default at 100%. This option is related to the frequent situation where several distinct trees are equally parsimonious. The dnapars and protpars methods, and thus Seaview, complete their analysis computing the consensus of all equally parsimonious trees found and present as result a single tree, the consensus, instead of a collection of equally optimal trees. Seaview computes by default the 100% consensus meaning that only internal branches present in 100% of most parsimonious trees are retained in the consensus. The frequency threshold required for an internal branch to appear in the consensus can be changed between 100% and the minimum value of 50%. Keeping the threshold above 50% ensures the existence of a single consensus tree without contradiction between alternative internal branch choices. Consensus trees are expected to be only partially resolved, that is, to contain multifurcations where individual most parsimonious trees propose distinct resolutions.

The parsimony algorithm can be applied under a bootstrap approach, which repeats the full search for the most parsimonious tree topology with the chosen number of bootstrap replicates. In that case, Seaview computes the bootstrap support of each branch of the tree computed from native sequences, which is the frequency at which that branch appears among trees computed from the bootstrap replicates.

For all bootstrap analyses, Seaview implements both the standard bootstrap introduced by Felsenstein [20], and a recently proposed modification of that approach called "Transfer Bootstrap" intended to behave better for datasets containing hundreds of sequences [21].

2.9 Distance-Based Tree Building

The distance-based approach to phylogenetic tree building proceeds by computing first pairwise phylogenetic distances between sequences, and then estimating the topology and the branch lengths of the tree that best represents these distances [22]. The phylogenetic distance between two homologous sequences is defined as the average number per sequence site of substitution or replacement events that occurred between these two sequences since divergence from their last common ancestor. The phylogenetic distance between two sequences is necessarily larger than the observed distance between them (number of observed differences / number of sites). If the evolutionary process is assumed to have followed a given probabilistic model at all sequence sites and all evolutionary dates, it can be possible to estimate phylogenetic

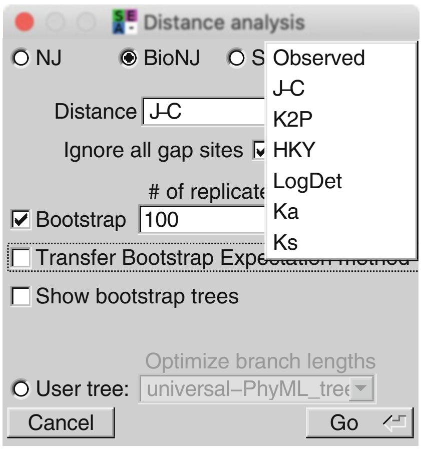


Fig. 8 The distance methods dialog window

distances from sequence data. Various probabilistic models of the evolutionary process have been proposed. Some of them are implemented in Seaview (Fig. 8).

The neighbor-joining method (NJ) is a widely used algorithm that constructs a tree topology and its branch lengths from the matrix of pairwise sequence distances [23]. This algorithm has been improved under the name BioNJ to take into account the fact that large evolutionary distances are less accurately estimated than short ones [24]. Both methods are implemented in Seaview (Fig. 8).

When applied to DNA sequences, Seaview can compute pairwise distances according to 7 approaches. (1) Observed distances defined above can be used when a very rough summary of sequence dissimilarity is sought. (2) The Jukes and Cantor (JC), (3) Kimura two-parameter (K2P), and (4) HKY models are standard probabilistic models of molecular evolution of increasing realism and have been summarized by Rzhetsky and Nei [25]. (5) LogDet was proposed by Lake [26] and Lockhart et al. [27]. The (6) Ka and (7) Ks methods aim at better dealing with the evolution of protein-coding DNA sequences. Therein, two distinct evolutionary processes are mixed, one producing synonymous substitutions, which don't change the encoded amino acid, and one producing non-synonymous substitutions that change the encoded amino acid. Synonymous substitutions typically occur at a higher rate in protein-coding sequences than non-synonymous ones because non-synonymous substitutions can be eliminated by natural selection if the resulting encoded protein has an adverse effect on the organism's phenotype. Li has proposed to compute two evolutionary distances for a pair of protein-coding DNA sequences, one called Ks quantifying the number of synonymous substitutions per synonymous sites, another called Ka quantifying the number of non-synonymous substitutions per non-synonymous sites

[28]. The Ks analysis provides more phylogenetic resolution for small evolutionary divergences. The Ka analysis allows to consider more divergent sequences for tree building.

When applied to protein sequences, Seaview can compute three kinds of pairwise distances: (1) observed, (2) Poisson, and (3) Kimura. Poisson distances assume all amino acid replacements occur at the same rate at all sequence sites. Kimura distances are based on an empirical approximation of the evolutionary distances proposed by Kimura (reviewed in [22]).

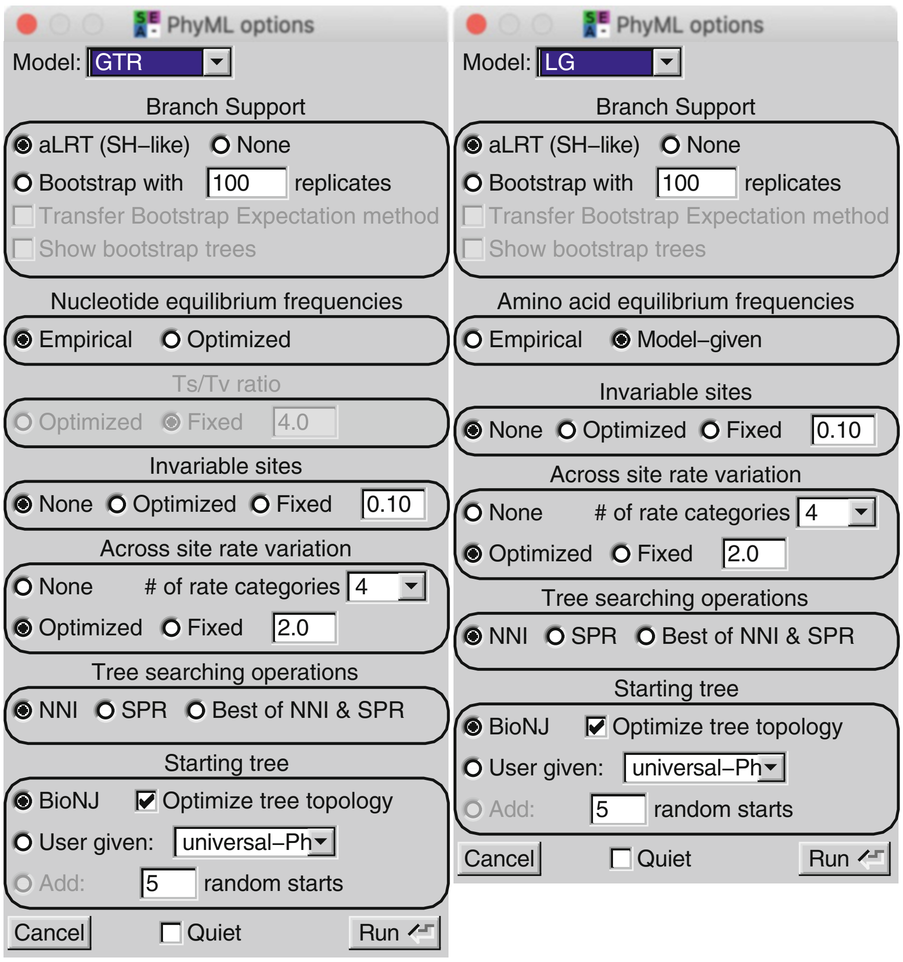
Altogether, all pairwise evolutionary distances computed by Seaview assume all sequence sites (separately for synonymous and non-synonymous sites) evolve at the same rate. Such assumption tends to underestimate evolutionary distances for large divergences [29], potentially biasing subsequent tree reconstructions. For this reason, it is recommended to prefer tree building by maximum likelihood and to use distance-based phylogenetic trees only as a rough estimation of sequence relationships.

Both NJ and BioNJ methods are very fast, so they can be combined with the bootstrap approach, which requires to apply the method to several hundred replicates of the sequence dataset.

2.10 Tree Building by Maximum Likelihood

The maximum likelihood criterion is a statistical concept of extremely wide applicability. It has been introduced by Felsenstein in the field of molecular phylogenetic reconstruction [30], and has become one of the most successful tree-building procedure. Tree building by maximum likelihood benefits of several key properties of the statistical theory which guarantee, for a wide set of scenarios, that the maximum likelihood tree converges to the true tree if the evolutionary process did follow the probabilistic model assumed by this method and if the length of the sequence dataset grows to infinity. The maximum likelihood approach to molecular phylogeny can be applied both to DNA and to protein sequences. Several implementations of that method have been developed during the last decade, such as PhyML [31], RaxML [32], and IQ-TREE [33]. Seaview allows to compute phylogenetic trees using the maximum likelihood approach with the PhyML version 3.1 program [34]. Other implementations may perform faster with large datasets, but all implement the same probabilistic model of molecular evolution and therefore are expected to converge to the same phylogenetic tree, supposing that the optimal tree is found.

Figure 9 shows the dialogs Seaview presents to pilot a PhyML run (left, DNA sequences; right, protein sequences). The "Model" pop-up menu allows to choose among various models, that is, relative probabilities of base substitutions or of amino acid replacements. For DNA sequences, the "general time reversible" (GTR) model is the most general of all models that can be used. It can accommodate unequal base compositions and biased transition over transversion rates and is the recommended model to choose for phylogenetic tree building by maximum likelihood [35].



PhyML options

Model: GTR

Branch Support

[✓ aLRT (SH–like)] None

Bootstrap with 100 replicates

Transfer Bootstrap Expectation method

Show bootstrap trees

Nucleotide equilibrium frequencies

[✓ Empirical] Optimized

Ts/Tv ratio

Optimized [✓ Fixed] 4.0

Invariable sites

[✓ None] Optimized Fixed 0.10

Across site rate variation

None # of rate categories 4

[✓ Optimized] Fixed 2.0

Tree searching operations

[✓ NNI] SPR Best of NNI & SPR

Starting tree

[✓ BioNJ] [✓ Optimize tree topology]

User given: universal–Ph

Add: 5 random starts

Cancel Quiet Run

PhyML options

Model: LG

Branch Support

[✓ aLRT (SH–like)] None

Bootstrap with 100 replicates

Transfer Bootstrap Expectation method

Show bootstrap trees

Amino acid equilibrium frequencies

Empirical [✓ Model–given]

Invariable sites

[✓ None] Optimized Fixed 0.10

Across site rate variation

None # of rate categories 4

[✓ Optimized] Fixed 2.0

Tree searching operations

[✓ NNI] SPR Best of NNI & SPR

Starting tree

[✓ BioNJ] [✓ Optimize tree topology]

User given: universal–Ph

Add: 5 random starts

Cancel Quiet Run

Fig. 9 Dialog windows to run PhyML. Left, with a DNA alignment; right, with a protein sequence alignment

For protein sequences, maximum likelihood algorithms use a precomputed matrix of relative rates of replacement between pairs of amino acids. Various authors have proposed such rate matrices since Dayhoff's pioneer computation starting from a few tens of families of homologous, closely related protein sequences [36]. Le and Gascuel have produced the most recent estimate of these relative replacement rates based on about 50,000 protein sequences from 3900 families spanning small and large divergence levels and accounting for the variability of replacement rates along proteins [37]. The LG matrix is therefore the recommended choice for the model parameter.

The branch support part of the dialog allows to enrich the resulting tree with estimates of the statistical support of each of its internal branches. That can be done quickly, with an approximation called aLRT, or in a much more time-consuming way, through bootstrap. Bootstrap results can be interpreted in the classical way

[20], or in a recently proposed alternative fashion called "Transfer bootstrap expectation" intended to be more efficient for large trees [21].

PhyML dialog parts labeled "Nucleotide equilibrium frequencies," "Ts/Tv ratio," and "Amino acid equilibrium frequencies" correspond to parameter values that are either set to observed values in the dataset, set to user-given values, or set to an optimal value estimated by the program. The "Invariable sites" part determines whether the model assumes there exists a fraction of invariable sites in the dataset, and if so, if that fraction is set a priori by the user or optimized by the program. The "Across site rate variation" part determines whether the model assumes all sites evolved with the same rate (None) or if sites are supposed to have a gamma-distributed evolutionary rate. That distribution itself is discretized in a number of rate categories (4 by default). The alpha parameter value of that distribution is either optimized by the program or set a priori by the user.

Three choices are possible in the "Tree searching operations" part which both increase computation time and decrease the probability for the algorithm to remain trapped in local optima [34]. The "Starting tree" part makes sense in that PhyML is an algorithm that begins with a starting tree, computes its likelihood, and keeps modifying branch lengths and topology of the current tree until no modification that improves the tree likelihood is found. The default starting tree is obtained by computing pairwise sequence distances and applying the BioNJ method to them. If the bottom of the "Trees" menu contains trees, any such tree can be used as starting point after selecting the "User given": option instead of "BioNJ." When" SPR" or "Best of NNI and SPR" is selected, a third way of defining the starting tree becomes possible: use a number (5 by default) of random starting trees, repeat the complete algorithm starting from each of those, and finally retain the tree with the highest likelihood found. Option "Optimize tree topology" can be turned off to ask PhyML to evaluate the likelihood of a user-given tree topology.

3 The Tree Window

When the computation of a phylogenetic tree by any of the methods mentioned above completes, Seaview presents the resulting tree in a tree window (Fig. 10). A tree window can also be created by opening a local file containing a tree in Newick format [18]. Once a tree window has been created, the user can select the "Save to Trees menu" item of the "File" menu to add that tree to the "Trees" menu of the corresponding alignment window. Consequently, that tree gets saved whenever the alignment data is written to a file. That is the recommended way to preserve the

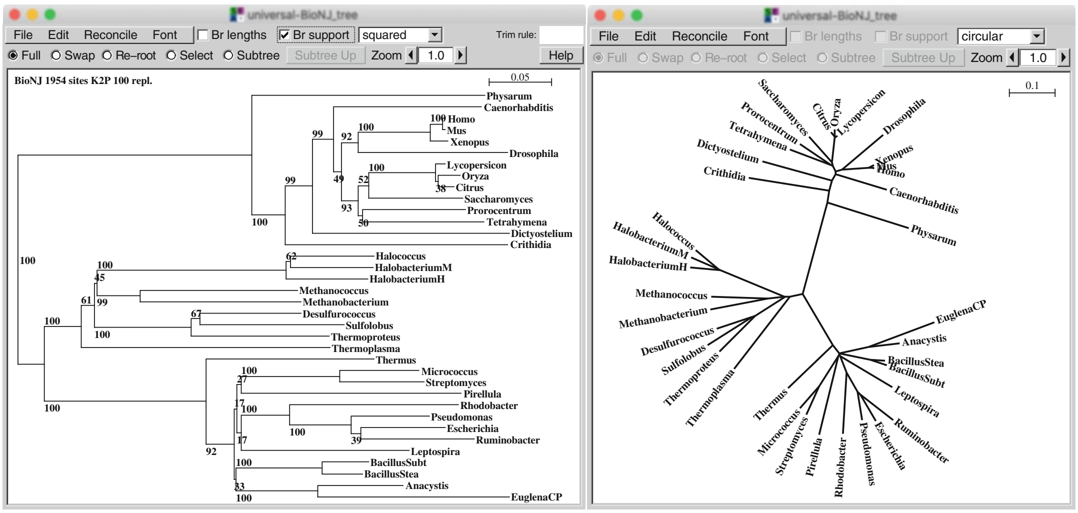


Fig. 10 Two forms of the Tree window. An example gene tree appears in “squared” (left) and “circular” (right) forms. Branch support levels as percent values resulting from a bootstrap analysis appear at left

result of a phylogenetic analysis performed with Seaview. The NEXUS and Mase file formats both allow to store a multiple sequence alignment and several corresponding trees in a single file.

Any tree window displays the tree with or without showing branch lengths or statistical levels of branch support. Buttons “Swap” and “Reroot” perform operations that change the graphical display of the tree but keep its topology unaltered. The “Select” button allows to select one or a cluster of sequences in the tree window and have the corresponding sequences selected in the alignment window. The menu item “Edit/Edit tree shape” allows to manipulate the tree topology itself performing subtree prune-and-regraft operations in a copy of a starting tree.

A pop-up menu with three items, “squared,” “circular,” and “cladogram,” allows to display the underlying phylogenetic tree in one of three different fashions: (1) “squared” shows it rooted and with branches proportional to branch lengths when these data are present in the underlying tree (Fig. 10, left); (2) “circular” draws the tree in a circular plot that aims at representing the abstract concept of an unrooted tree; and (3) “cladogram” draws the tree in a rooted form but with all root-to-leave distances equal with the aim to represent the abstract concept of a tree without branch length information.

Phylogenetic trees can be rooted or unrooted, binary or multifurcated, with or without branch lengths, with or without indication of statistical support of their internal branches. The Newick format [18] can represent all these sorts of trees, and Seaview can draw all of them. Phylogenetic trees computed with Seaview, except reconciled trees (*see* below), have these properties: they are unrooted, without multifurcation, and all but parsimony trees on protein data have branch lengths. All can optionally be computed under a bootstrap analysis, which provides statistical support values of all internal branches expressed as percent values. The PhyML method additionally offers the option to estimate the statistical support of internal branches by the aLRT method expressed as a number in the [0–1] range. When Seaview draws a rooted tree using the "squared" type of tree display, it uses the root location indicated by the tree. When it draws an unrooted tree, Seaview tentatively places the root in the tree with the midpoint method, that is, at the center of the longest path between two tree leaves, as measured by adding the lengths of branches connecting these leaves. When Seaview draws an unrooted tree without branch length, it roots it arbitrarily on one of its terminal branches. Seaview users are strongly advised to use the "Reroot" button of tree windows and reconsider under the light of their biological knowledge the tentative rooting of any unrooted tree proposed by Seaview. Alternatively, they can attempt a gene tree/species tree reconciliation if the tree under consideration is a gene tree and if a species tree is available, which can root the gene tree (*see* the "Reconcile" menu described below). An unrooted tree can be saved to a rooted form and in Newick format with item "Save rooted tree" of the "File" menu of any tree window. It can also be saved to an unrooted form through item "Save unrooted tree."

Items "Print," "Save as PDF," and "Save as SVG" of the "File" menu of tree windows allow to store the current tree in its graphical form. Noticeably, the output of the first two items for "squared" trees can be made to cover a user-chosen number of pages, which support the display of trees containing hundreds of leaves.

The "Edit/Copy" menu item of tree windows copies the current tree in its current shape (rooted, circular, cladogram) to the clipboard. That graphics can then be pasted to any adequate drawing application, for example, to prepare a figure for publication. Under Linux/Unix, it may be more convenient to save the tree in SVG form using the "File/Save as SVG" menu item and to process the resulting SVG file, for example, with the Inkscape application.

The "Reconcile" menu is new in Seaview version 5. It allows to compare a gene tree (possibly unrooted and multifurcated) with a species tree (which currently has to be rooted and fully resolved) to perform three tasks: (1) reconcile them, that is, annotate the gene tree nodes with speciation, duplication, and loss events, and compute the reconciliation score, which is the number of duplications and losses, multiplied by their respective costs, set by the user in a parameter window; (2) root the gene tree with a root that minimizes the reconciliation score (when roots with equal scores exist, a randomly chosen one is used); (3) and rearrange the gene tree branches with low statistical support (what is "low" is a user-defined parameter, often set to 80% for bootstrap supports). The

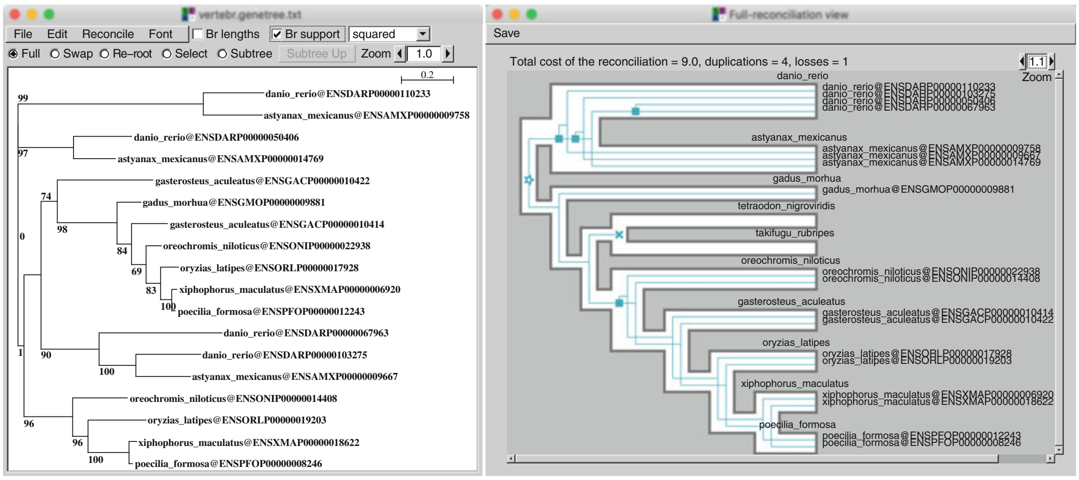

vertebr.genetree.txt
File Edit Reconcile Font
Br lengths
Br support
squared
Full
Swap
Re–root
Select
Subtree
Subtree Up
Zoom 1.0
0.2
99
danio_rerio@ENSDARP00000110233
astyanax_mexicanus@ENSAMXP00000009758
danio_rerio@ENSDARP00000050406
97
astyanax_mexicanus@ENSAMXP00000014769
gasterosteus_aculeatus@ENSGACP00000010422
74
gadus_morhua@ENSGMOP00000009881
98
gasterosteus_aculeatus@ENSGACP00000010414
0
84
oreochromis_niloticus@ENSONIP00000022938
69
oryzias_latipes@ENSORLP00000017928
83
xiphophorus_maculatus@ENSXMAP00000006920
100
poecilia_formosa@ENSPFOP00000012243
danio_rerio@ENSDARP00000067963
1
90
danio_rerio@ENSDARP00000103275
100
astyanax_mexicanus@ENSAMXP00000009667
oreochromis_niloticus@ENSONIP00000014408
96
oryzias_latipes@ENSORLP00000019203
96
xiphophorus_maculatus@ENSXMAP00000018622
100
poecilia_formosa@ENSPFOP00000008246
Full-reconciliation view
Save
Total cost of the reconciliation = 9.0, duplications = 4, losses = 1
1.1
Zoom
danio_rerio
danio_rerio@ENSDARP00000110233
danio_rerio@ENSDARP00000103275
danio_rerio@ENSDARP00000050406
danio_rerio@ENSDARP00000067963
astyanax_mexicanus
astyanax_mexicanus@ENSAMXP00000009758
astyanax_mexicanus@ENSAMXP00000009667
astyanax_mexicanus@ENSAMXP00000014769
gadus_morhua
gadus_morhua@ENSGMOP00000009881
tetraodon_nigroviridis
takifugu_rubripes
oreochromis_niloticus
oreochromis_niloticus@ENSONIP00000022938
oreochromis_niloticus@ENSONIP00000014408
gasterosteus_aculeatus
gasterosteus_aculeatus@ENSGACP00000010414
gasterosteus_aculeatus@ENSGACP00000010422
oryzias_latipes
oryzias_latipes@ENSORLP00000017928
oryzias_latipes@ENSORLP00000019203
xiphophorus_maculatus
xiphophorus_maculatus@ENSXMAP00000006920
xiphophorus_maculatus@ENSXMAP00000018622
poecilia_formosa
poecilia_formosa@ENSPFOP00000012243
poecilia_formosa@ENSPFOP00000008246

Fig. 11 A gene tree/species tree reconciliation. Left: a gene tree taken from Ensembl [39] showing the statistical support of each branch expressed in per cent. Right: this gene tree has been reconciled with the species tree, also taken from Ensembl, using the Treerecs method, and the reconciled gene tree is drawn, using Seaview, inside the species tree. The threshold for branch support has been set to 99%: some branches of the gene tree with support <99% have been rearranged in the reconciled tree to minimize the reconciliation score. The reconciled tree predicts four gene duplication events (solid squares) and one gene loss (cross). The star denotes the root of the species tree

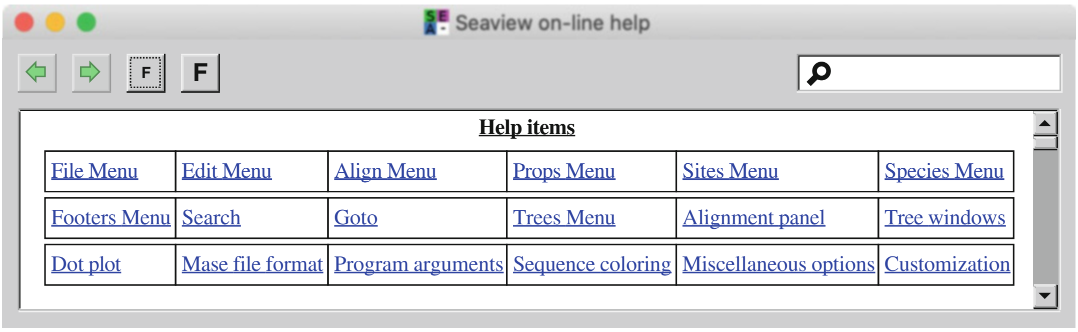
latter option means that a new gene tree is constructed, which contains all the branches of the initial gene tree when they have a high support, and the other branches are constructed to minimize the reconciliation score. This can be performed simultaneously with the rooting or with a specified root. Multifurcated gene trees can also be resolved as binary trees following their reconciliation with the species tree. All these operations are performed with the Treerecs method [38], which runs a dynamic programming algorithm along the nodes of the gene and species trees, in order to map and resolve gene tree nodes with a parsimony objective (minimizing the reconciliation score).

Reconciliations imply that each gene (gene tree leaf) is mapped onto the species (species tree leaf) it belongs to. Treerecs automatically detects this mapping if the species name is a part of the gene name. Otherwise, a mapping file, containing lines with gene-species correspondences, is required.

Reconciliations can be visualized in an additional window, as shown in Fig. 11.

4 The Help Window

Help information detailing the purpose of all Seaview features and how to trigger them is available while running the program in the form of a window containing an HTML document. This



Help items

File Menu	Edit Menu	Align Menu	Props Menu	Sites Menu	Species Menu
Footers Menu	Search	Goto	Trees Menu	Alignment panel	Tree windows
Dot plot	Mase file format	Program arguments	Sequence coloring	Miscellaneous options	Customization

Fig. 12 The top of the Help window lists 18 sections covering all Seaview uses

information is divided in 18 sections that cover all program functions (Fig. 12). That document is also readable with any web browser.

5 Command-Line Mode

Seaview can also be used in command-line mode, that is, as a command line composed of the path to the Seaview executable on the system and a series of arguments defining an operation to be performed by Seaview. In that mode, no graphical user interface appears, which allows Seaview to be usable in user-written scripts where the same task is repeated a number of times. Most of the functions described above can also be performed in command-line mode. Tree building by PhyML and reconciliation by Treerecs, though, are not covered by Seaview's command-line mode because the PhyML and Treerecs programs themselves natively run as a command line. One command line to perform a multiple sequence alignment could be, for example:

```
/path/to/seaview -align -align_algo 1 -align_at_protein_level
-output_format phylip -o outfile.phy infile.nxs
```

The arguments in that command line successively indicate to perform a multiple sequence alignment using algorithm #1, which is muscle, to translate input sequences to protein, align them, and then transfer the protein-level alignment to the DNA-level, to output the resulting alignment as a PHYLIP-formatted file and name the output (outfile.phy) and input (infile.nxs) files. The seven groups of operations Seaview can perform in command-line mode are summarized in Table 1.

The "Program arguments" section of the Seaview help window (Fig. 12) details all program arguments usable in command-line mode.

Table 1
Groups of operations available in command-line mode

Option name	Option use
– convert	Convert an input alignment to another file format
– concatenate	Concatenate several alignments
– align	Perform multiple sequence alignment
– build_tree	Compute a parsimony- or distance-based phylogenetic tree
– printout	Draw a multiple sequence alignment to PDF or SVG format
– reroot	Modify the rooting of an input tree
– plotonly	Draw a tree to a PDF or SVG file

Acknowledgments

We thank the authors of all software bundled with Seaview (Clustal, muscle, gblocks, PhyML, PHYLIP, Treerecs) for permission to use these fine products. We thank Frédéric Lemoine and Olivier Gascuel for contributing code for the “Transfer bootstrap.”

References

1. Phylogeny Programs. http://evolution.genetics.washington.edu/phylip/software.html
2. Gouy M, Guindon S, Gascuel O (2010) SeaView version 4: a multiplatform graphical user interface for sequence alignment and phylogenetic tree building. Mol Biol Evol 27:221–224
3. International Nucleotide Sequence Database Collaboration|INSDC, http://www.insdc.org/
4. PRABI-Doua: SeaView, http://doua.prabi.fr/software/seaview
5. Debian—Details of package seaview in buster, https://packages.debian.org/buster/seaview
6. PHYLIP Home Page. http://evolution.gs.washington.edu/phylip.html
7. 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
8. Maddison DR, Swofford DL, Maddison WP (1997) NEXUS: an extensible file format for systematic information. Syst Biol 46:590–621
9. Wilgenbusch JC, Swofford D (2003) Inferring evolutionary trees with PAUP*. Curr Protoc Bioinforma 6(4):1–6.4.28
10. Castresana J (2000) Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Mol Biol Evol 17:540–552
11. Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32:1792–1797
12. Katoh K, Misawa K, Kuma K et al (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res 30:3059–3066
13. 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
14. Roshan U (2014) Multiple sequence alignment using probcons and probalign. In: Russell D. (eds) Multiple sequence alignment methods. Methods in Molecular Biology (Methods and Protocols), vol 1079. Humana Press, Totowa, NJ. https://doi.org/10.1007/978-1-62703-646-7_9
15. Ranwez V, Harispe S, Delsuc F et al (2011) MACSE: multiple alignment of coding SEquences accounting for frameshifts and stop codons. PLoS One 6:e22594

16. Jukes TH, Osawa S (1993) Evolutionary changes in the genetic code. Comp Biochem Physiol B 106:489–494
17. Osawa S, Jukes TH, Watanabe K et al (1992) Recent evidence for evolution of the genetic code. Microbiol Rev 56:229–264
18. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html
19. Hochbaum DS, Pathria A (1997) Path costs in evolutionary tree reconstruction. J Comput Biol 4:163–175
20. Felsenstein J (1985) Confidence limits on phylogenies: an approach using the bootstrap. Evol Int J Org Evol 39:783–791
21. Lemoine F, Domelevo Entfellner J-B, Wilkinson E et al (2018) Renewing Felsenstein's phylogenetic bootstrap in the era of big data. Nature 556:452–456
22. Felsenstein J (2003) Inferring phylogenies, Sinauer Associates, Sunderland MA
23. Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4:406–425
24. Gascuel O (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Mol Biol Evol 14:685–695
25. Rzhetsky A, Nei M (1995) Tests of applicability of several substitution models for DNA sequence data. Mol Biol Evol 12:131–151
26. Lake JA (1994) Reconstructing evolutionary trees from DNA and protein sequences: paralinear distances. Proc Natl Acad Sci U S A 91:1455–1459
27. Lockhart PJ, Steel MA, Hendy MD et al (1994) Recovering evolutionary trees under a more realistic model of sequence evolution. Mol Biol Evol 11:605–612
28. Li WH (1993) Unbiased estimation of the rates of synonymous and nonsynonymous substitution. J Mol Evol 36:96–99
29. Tourasse NJ, Gouy M (1999) Accounting for evolutionary rate variation among sequence sites consistently changes universal phylogenies deduced from rRNA and protein-coding genes. Mol Phylogenet Evol 13:159–168
30. Felsenstein J (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol 17:368–376
31. Guindon S, Gascuel O (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst Biol 52:696–704
32. Stamatakis A, Ludwig T, Meier H (2005) RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinforma Oxf Engl 21:456–463
33. Nguyen L-T, Schmidt HA, von Haeseler A et al (2015) IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol Biol Evol 32:268–274
34. Guindon S, Dufayard J-F, Lefort V et al (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. Syst Biol 59:307–321
35. Abadi S, Azouri D, Pupko T et al (2019) Model selection may not be a mandatory step for phylogeny reconstruction. Nat Commun 10:934
36. Dayhoff MO (1969) Atlas of protein sequence and structure. National Biomedical Research Foundation
37. Le SQ, Gascuel O (2008) An improved general amino acid replacement matrix. Mol Biol Evol 25:1307–1320
38. Comte N, Morel B, Hasic D et al (2020) Treerecs: an integrated phylogenetic tool, from sequences to reconciliations. Bioinformatics https://doi.org/10.1093/bioinformatics/btaa615
39. Yates AD, Achuthan P, Akanni W et al (2020) Ensembl 2020. Nucleic Acids Res 48: D682–D688

Chapter 16

NCBI Genome Workbench: Desktop Software for Comparative Genomics, Visualization, and GenBank Data Submission

Anatoliy Kuznetsov and Colleen J. Bollin

Abstract

The book chapter introduces the National Center for Biotechnology Information (NCBI) Genome Workbench, a desktop GUI software package to manipulate and visualize complex molecular biology models provided in many data formats. Genome Workbench integrates graphical views and computational tools in a single package to facilitate discoveries. In this chapter we provide a step-by-step protocol guidance on how to do comparative analysis of sequences using NCBI BLAST and multiple sequence alignment algorithms, build phylogenetic trees, and use graphical views for sequences, alignments, and trees to validate the findings. The software package can be used to prepare high-quality whole genome submissions to NCBI. The software package is user-friendly and includes validation and editing tools to fix errors as part of preparing the submission.

Key words Phylogenetic, Alignment, MUSCLE, MAFFT, Clustal, Analysis, GenBank, Genome, PubMed, BLAST, Visualization, Bioinformatics

1 NCBI Genome Workbench: Capabilities Overview

1.1 Introduction

NCBI Genome Workbench is a set of desktop GUI tools for studying genetic data. It can connect to NCBI data sources or load local, private data files. The software uses a set of graphical displays called Views to display genomic data. Genome Workbench offers seamless integration with NCBI and non-NCBI algorithmic tools to run analysis. The process of scientific discovery involves making connections between unobvious observations coming from comparisons between public and private datasets. Genome Workbench offers integrated tools to quickly run algorithms, experiment with parameters, and evaluate results using graphical views without engaging with heavy duty bioinformatics pipelines or programming. Genome Workbench also offers tools to prepare full genome submissions to NCBI.

1.2 Download and Install Genome Workbench

The main URL for software and documentation is https://www.ncbi.nlm.nih.gov/tools/gbench/

The software suite is open source and multiplatform. NCBI supports Microsoft Windows, MacOS, and variants of Linux. The user must pick the right distribution package, download it, and install it on their PC, Mac or Linux. Some Linux users may find it best to use source code distribution to build their own for Linux distributions not supported by NCBI.

1.3 Data Import Capabilities

Genome Workbench can load data from several bioinformatics formats and data sources, such as FASTA, NCBI ASN.1 (text or binary), BAM/cSRA (local databases or remote http hosted data sources), BED, GFF, VCF, NEWICK, NEXUS, RepeatMasker, FASTA alignments, Wiggle, and 5-column feature table.

Genome Workbench provides the ability to search and load data from NCBI in several ways:

1. Load genome collections (detailed description of sequence molecules which constitute a genome).
2. Load individual molecules using their NCBI accessions.
3. Discover annotations for molecules.
4. Load results of BLAST alignment algorithms by their RID (special token issued when a BLAST request is submitted).

1.4 Data Privacy

Genome Workbench is a desktop tool working on the user's computer with the user's own data. Sequence and annotation data are processed locally on the user's computer. It is not sent over the network, unless the user explicitly requests this. This offers greater privacy control than using web genomics tools.

For quality monitoring, Genome Workbench sends some information about usage statistics back to NCBI. This information is limited to names of the tools and views without any data association to reveal the potentially sensitive context of the user's research. It is possible to opt out and completely silence usage reporting.

1.5 Data Visualization

Genome Workbench implements a set of graphical displays called views to make data exploration easier.

The most important and useful Views of NCBI Genome Workbench are:

1. Text View—this is a generic, non-graphical view which allows the user to see DNA, RNA, or protein molecule information and annotations. Text View supports various formats of data presentation, where the most important is the GenBank Flat File Format, which displays molecule meta-information, annotated features, and sequence data.

2. Graphical Sequence View—this view uses a graphical model of the molecule, with zoom and scroll to see various types of annotations as tracks (feature tracks, graphs, SNPs, alignments, etc.). All track data are displayed by default in the molecular biology directionality system of 5′-to-3′, and can be reversed. Graphical Sequence View is designed to be fully interactive and display translation details, single nucleotide polymorphism mutations, and fine gene structure of exons and introns, making it easier to visually correlate various sources of scientific evidence and compare reference data with the details of a particular experiment.
3. Multiple Sequence Alignment Viewer—this graphical view is for comparative analysis of molecules. Multiple Sequence Alignments prepared by algorithmic tools can be displayed in a common system of coordinates, helping to better understand the relationships between sequences. MSA View is fully interactive and can project some annotations as displayed in Graphical Sequence View into alignment coordinates. This unique feature helps to make exploratory connections between the mathematical model of alignment and functional genomics, known from experiments.
4. Sequence Text View—this is a graphical view to see sequences and translation details.
5. Tree View—this is a graphical view to explore phylogenetic trees prepared by tree reconstruction algorithms or multiple alignment tools.

1.6 Data Presentation and High-Quality Printing

Conventional screenshots, even on high DPI resolution, do not provide the printing quality of scalable vector graphics.

NCBI Genome Workbench allows the user to export graphical data into vector format (PDF) for printing. Generated PDFs are made of vector graphics. Output vector formats are scalable and can be edited using third-party graphical software packages to produce graphics for high-quality poster printing (menu File/Save As PDF).

1.7 Tools

Genome Workbench offers integration with an array of tools to help researchers.

1. BLAST Family of Tools [1, 2]—using Genome Workbench, the user is able to submit a BLAST search to NCBI and load results or run BLAST locally as pairwise or run against a local BLAST database.
2. Alignment Clean Up Tools—complementary to BLAST to reduce noise in the original alignment to see the big picture using graphical views.
3. Splign and Protein Splign—tools for calculating spliced alignments for fine gene structure.

4. Phylogenetic Tree Builder.
5. Multiple Alignment Tools—Genome Workbench integrates a number of external tools to help build multiple alignments. External, non-NCBI tools do not come with Genome Workbench; the user must download and install them separately. Supported tools for multiple alignments: MUSCLE [3], MAFFT, Kalign, and Clustal [4 (Chapter 10)].

1.8 Genomic Data Editing Package

Genome Workbench is modular software; it comes with extended functionality packages which can be enabled. The Editing Package extends the functionality of Genome Workbench to help the user prepare NCBI data submissions. When enabled, the Editing Package changes the Genome Workbench system of menus and available dialogs to allow data editing. With Genome Workbench, the user can import sequence data and annotations, run validations, and fix logical errors.

2 Getting Started with Genome Workbench

2.1 Main Window Look and Feel: Loading Data into the Project

When Genome Workbench starts up, the Main Window appears with several different panes.

The Project Tree View is on the left and is empty upon startup. This is where loaded data, analysis results, and the views created will be stored.

The concept of a Project is important to understand when using Genome Workbench. A Project is defined as the sequence data, annotations, and alignments relevant to the user's work. By placing data in a single project, the user defines a coherent dataset, which can be displayed by the Views. A Project can be persistently saved to disk, to be restored later or shared with colleagues.

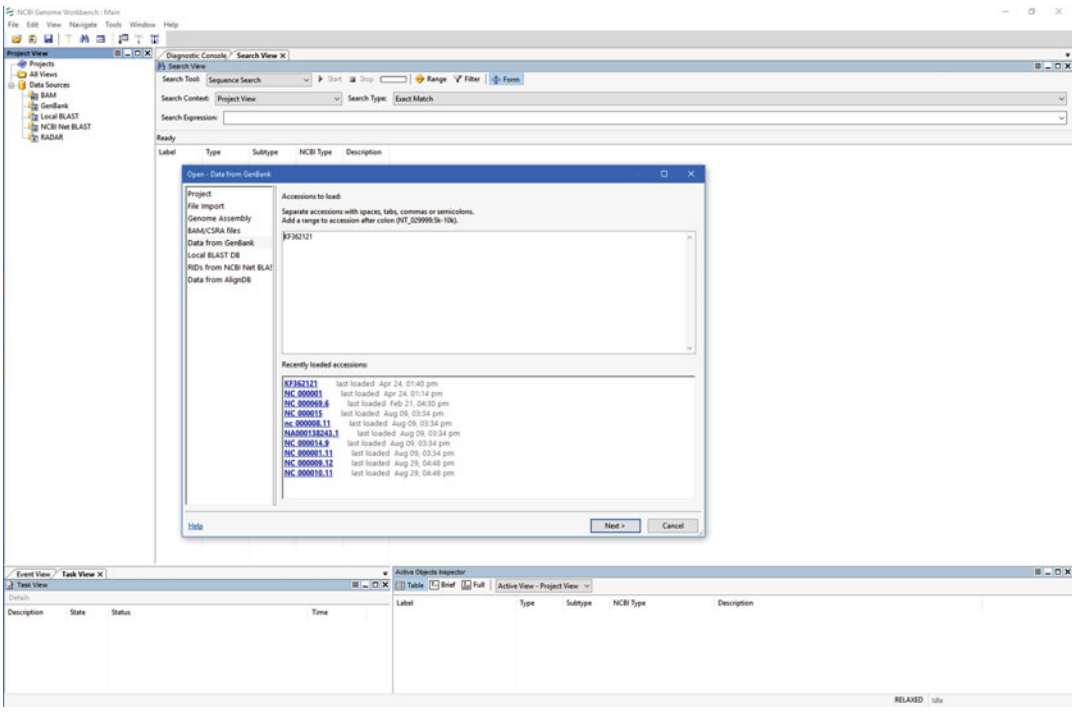
The Views (as referred to in this and other tutorials) are different windows within the Genome Workbench application providing the user with the information on various aspects of the work of the application and data. Views provided by the application are available through the View drop-down menu. Data views are opened by the user.

There are several ways to add data to a project.

To start with an NCBI GenBank [5] or RefSeq [6] accession, one can use Open Dialog to load the molecule (Fig. 1).

Accessions are special unique identifiers of molecules. There are a few patterns of accession assignment that may be useful to keep in mind:

- **AC**_complete genomic molecule, usually alternate assembly.
- **NC**_are reference sequence chromosomes, usually reference assembly.



Search Tool: Sequence Search

Search Context: Project View

Search Type: Exact Match

Search Expression:

Ready

Accessions to load:

Separate accessions with spaces, tabs, commas or semicolons.
Add a range to accession after colon (NT_029999:5k-10k).

KF362121

Recently loaded accessions:

```
KF362121        last loaded  Apr 24, 01:40 pm
NC_000001        last loaded  Apr 24, 01:14 pm
NC_000069.6       last loaded  Feb 21, 04:30 pm
NC_000015        last loaded  Aug 09, 03:34 pm
nc_000008.11      last loaded  Aug 09, 03:34 pm
NA000138243.1      last loaded  Aug 09, 03:34 pm
NC_000014.9       last loaded  Aug 09, 03:34 pm
NC_000001.11       last loaded  Aug 09, 03:34 pm
NC_000009.12       last loaded  Aug 29, 04:48 pm
NC_000010.11       last loaded  Aug 29, 04:48 pm
```

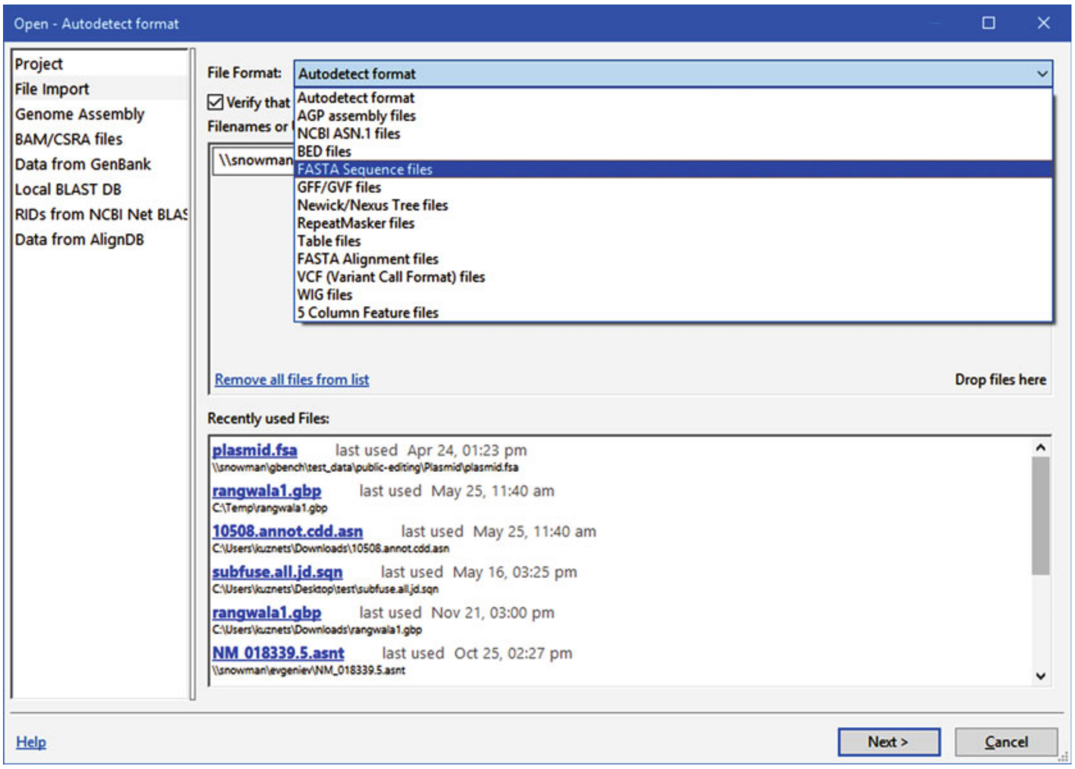
Fig. 1 Main Genome Workbench Windows with Open Dialog (GenBank data)

- **NT**_are reference sequence contigs or scaffolds, clone based, or WGS.
- **NW**_contig or scaffold, primarily WGS.
- **NG**_reference sequences that have had some degree of human curation, incomplete genomic region.
- **NZ**_complete genomes and unfinished WGS data.
- **NM**_reference sequence mRNAs.
- **NR**_non-protein-coding RNAs.
- **NP**_reference sequence proteins, associated with NM_ or NC_ accessions.
- **XM**_predicted model protein-coding mRNA.
- **XR**_predicted model non-protein-coding RNA.
- **XP**_predicted protein, associated with an XM_ accession.
- **YP**_protein annotated on genomic molecules without an instantiated

  transcript record.
- **WP**_protein, nonredundant across multiple strains and species.

Read more in the NCBI Hand Book, Chapter 18, The Reference Sequence (RefSeq) Database.

https://www.ncbi.nlm.nih.gov/books/NBK21091/



Recently used Files:

plasmid.fsa last used Apr 24, 01:23 pm
\\snowman\gbench\test_data\public-editing\Plasmid\plasmid.fsa
rangwala1.gbp last used May 25, 11:40 am
C:\Temp\rangwala1.gbp
10508.annot.cdd.asn last used May 25, 11:40 am
C:\Users\kuznets\Downloads\10508.annot.cdd.asn
subfuse.all.jd.sqn last used May 16, 03:25 pm
C:\Users\kuznets\Desktop\test\subfuse.all.jd.sqn
rangwala1.gbp last used Nov 21, 03:00 pm
C:\Users\kuznets\Downloads\rangwala1.gbp
NM 018339.5.asnt last used Oct 25, 02:27 pm
\\snowman\evgeniev\NM_018339.5.asnt

Fig. 2 Open Dialog for file import. It supports number of popular bioinformatics formats

2.2 Loading External (Non-NCBI) Annotations

External annotations can be loaded using the Open Dialog (File/ Open menu Fig. 2). Genome Workbench recognizes multiple bioinformatics formats. When using Open Dialog, one can allow Genome Workbench to auto-detect format or override the detection mechanism by explicit choice. Usually formats can be identified unambiguously, but there are a few cases when the detection algorithm may actually be incapable of choosing the format.

One of these cases is loading FASTA alignments, where the format is sometimes indistinguishable from FASTA sequence data, so manual format selection may be necessary.

Note that the Project View panel shows a list of various data items loaded into a project (Fig. 3).

Important! When loading data for the same genome, for example, FASTA of an organism and its GFF3 annotation—load it into the same project. If the annotation refers to the same sequence ids as listed in FASTA file—Genome Workbench will be able to associate features and sequence and display it graphically within the same coordinates on the genome.

2.3 Graphical Sequence View

To open a View, first select an object, in this case, a sequence, then launch the View dialog (Fig. 4).

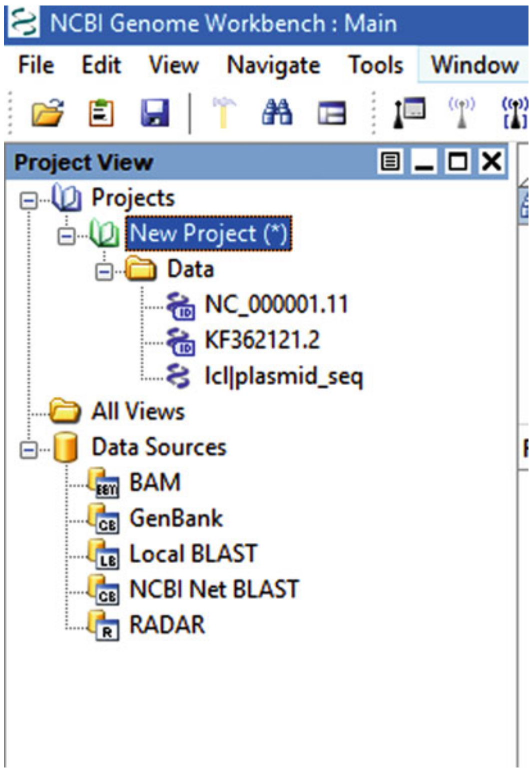


Fig. 3 Project View panel shows data loaded into a project

The View dialog has an option to set the default view associated with a particular data type (Fig. 5). Once a default view has been set, the user can use double-click in the Project View to launch it.

The visual language of the graphical view may seem complex. More information and legend information are available at:

https://www.ncbi.nlm.nih.gov/tools/sviewer/legends/

Please note, that this graphical notation is common between desktop graphical views of Genome Workbench and NCBI Web Genome Data Viewers.

Data of different types from different sources can be displayed graphically as "Tracks." (Fig. 6) The Graphical Sequence View supports different types of tracks for displaying sequence, 6-frame translation, genes (from different data sources), SNPs (all or categorized by clinical significance), structural variations, genome graphs (expression, epigenomic data, RNA-seq, ChIP-seq data, etc.), and alignments (from BAM files or after alignment programs like BLAST).

All Tracks support the concept of variable level of detail, depending on the zoom factor (Fig. 7).

As the user zooms in, more details will become visible. In Fig. 7, the view is zoomed all the way in to the actual sequence. The easiest and the most convenient way to zoom in and out is to

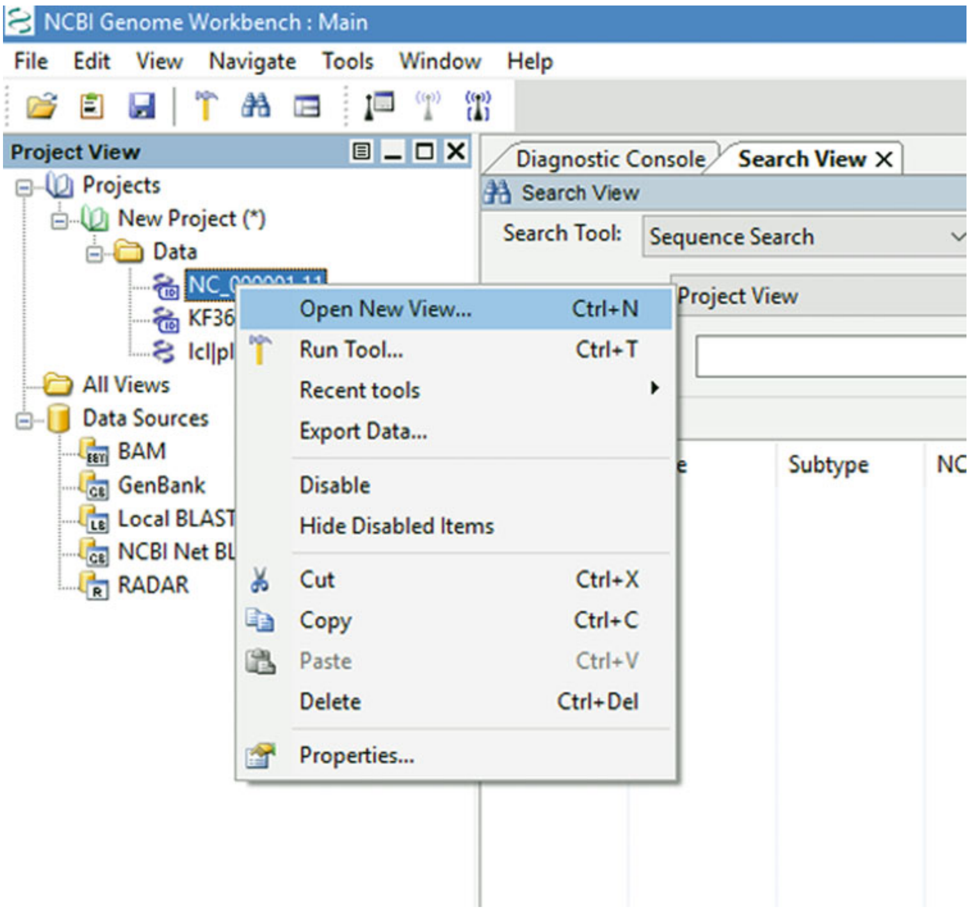


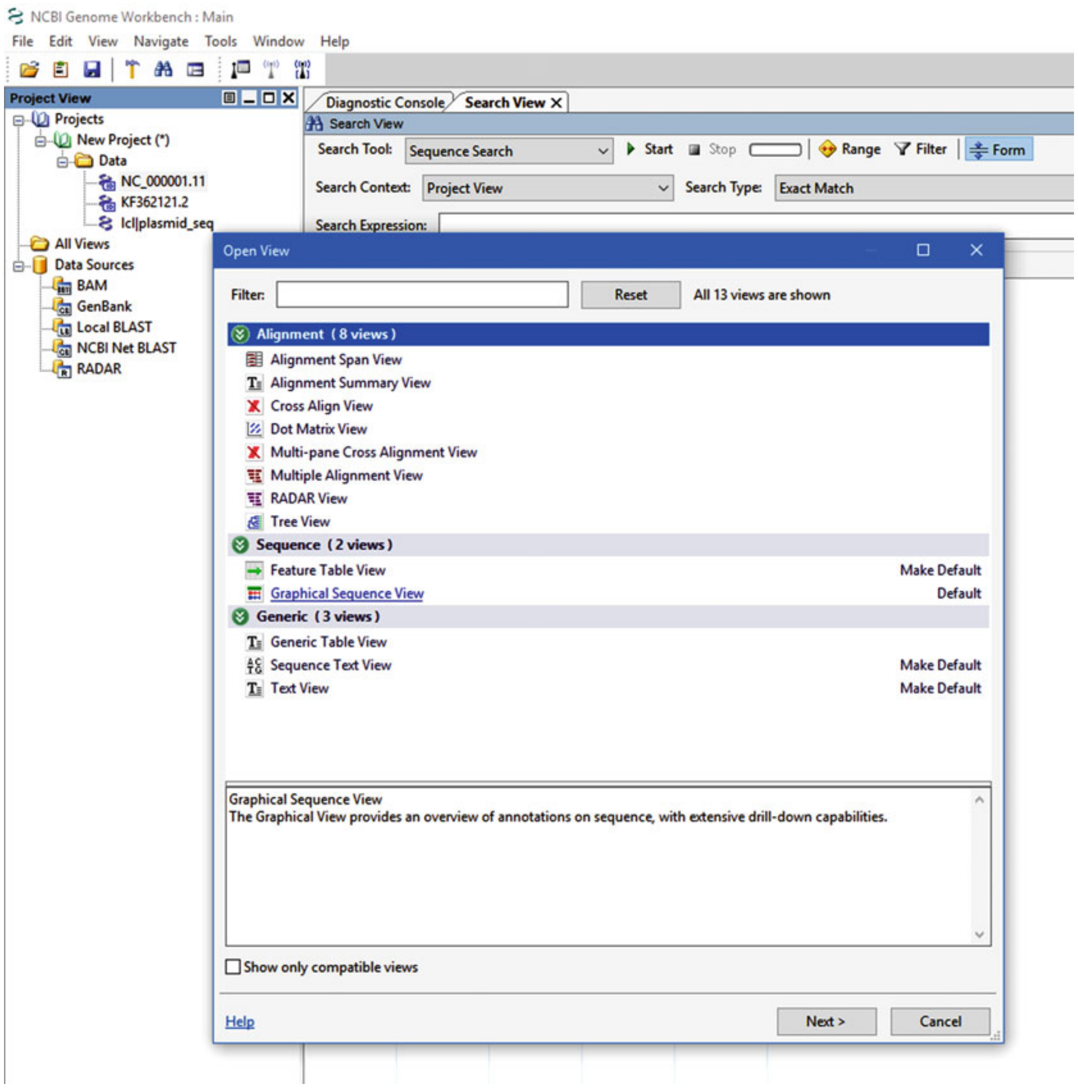
Fig. 4 Context menu (right mouse click) to open a View

use the mouse: position the mouse pointer at the zoom focus point on the sequence, press the mouse scroll wheel down and hold, and use the mouse to zoom in and out. The alternative is to hold "Z" button on the keyboard and hold the mouse button to zoom.

When zoomed in to the sequence level, the gray sequence bars across the top are now duplicated, showing both the forward and reverse (complemented) strand of the chromosomal sequence. In addition, inside the protein-coding region the letters of the protein are inscribed and spaced out to account for the codon boundaries and reading frame.

If the user selects a coding region feature, the letters of the codon responsible for the amino acid will be displayed beneath that amino acid.

When hovering over annotation or over the blue sequence bars, a tool tip will be displayed providing additional information. The images below show two tool tips: one over a feature, showing information about the feature as well as the GenBank display for



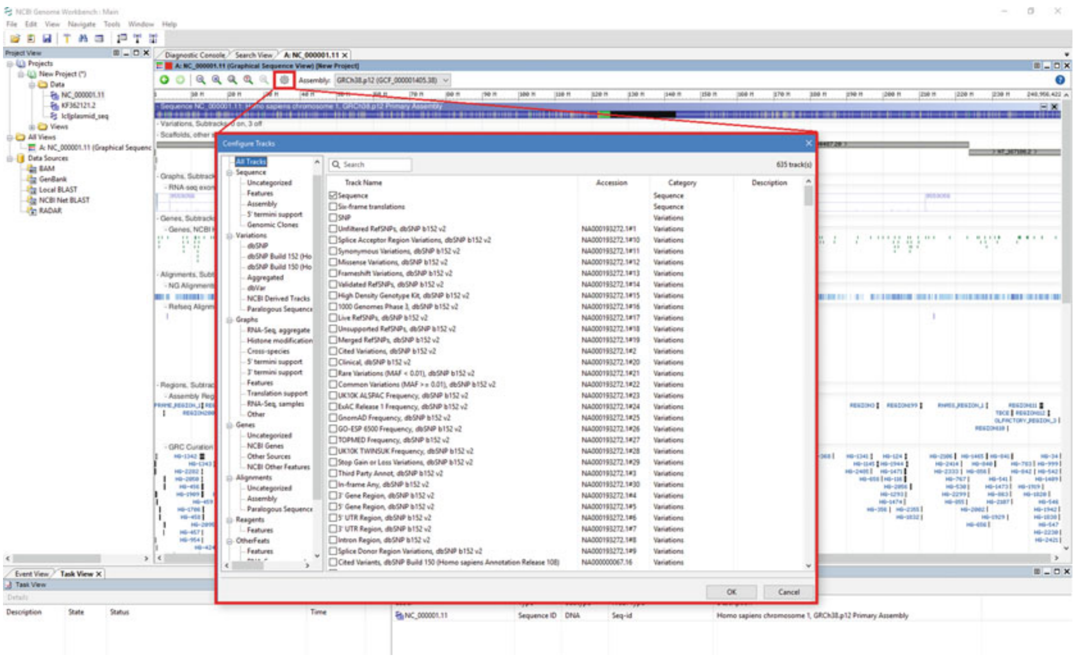
Graphical Sequence View
The Graphical View provides an overview of annotations on sequence, with extensive drill-down capabilities.

Fig. 5 Open View dialog, Graphical Sequence View is chosen as a default

this feature, and one for the sequence, providing details both about the organism involved and the location over which the mouse is positioned.

Graphical view also supports selections, which is a convenient way of specifying a molecule region for running a tool.

Selections can be tracked using the Active Object Inspector (Fig. 8). The Inspector can be configured to show a list of selected objects and molecule regions in the current View, or in all open Views. The screenshot shows objects selected in the Project and Graphical Sequence Viewer. Please note, that the Active Object Inspector shows both the selected region and the main molecule.


NCBI Genome Workbench : Main
File Edit View Navigate Tools Window Help
Project View
Diagnostic Console / Search View / A: NC_000001.11
Assembly: GRCh38.p12 (GCF_000001405.38)
Configure Tracks
All Tracks
Search
635 track(s)
OK
Cancel

Track Name	Accession	Category	Description
☑ Sequence		Sequence	
☐ Six-frame translations		Sequence	
☐ SNP		Variations	
☐ Unfiltered RefSNPs, dbSNP b152 v2	NA000193272.1#1	Variations	
☐ Splice Acceptor Region Variations, dbSNP b152 v2	NA000193272.1#10	Variations	
☐ Synonymous Variations, dbSNP b152 v2	NA000193272.1#11	Variations	
☐ Missense Variations, dbSNP b152 v2	NA000193272.1#12	Variations	
☐ Frameshift Variations, dbSNP b152 v2	NA000193272.1#13	Variations	
☐ Validated RefSNPs, dbSNP b152 v2	NA000193272.1#14	Variations	
☐ High Density Genotype Kit, dbSNP b152 v2	NA000193272.1#15	Variations	
☐ 1000 Genomes Phase 3, dbSNP b152 v2	NA000193272.1#16	Variations	
☐ Live RefSNPs, dbSNP b152 v2	NA000193272.1#17	Variations	
☐ Unsupported RefSNPs, dbSNP b152 v2	NA000193272.1#18	Variations	
☐ Merged RefSNPs, dbSNP b152 v2	NA000193272.1#19	Variations	
☐ Cited Variations, dbSNP b152 v2	NA000193272.1#2	Variations	
☐ Clinical, dbSNP b152 v2	NA000193272.1#20	Variations	
☐ Rare Variations (MAF < 0.01), dbSNP b152 v2	NA000193272.1#21	Variations	
☐ Common Variations (MAF >= 0.01), dbSNP b152 v2	NA000193272.1#22	Variations	
☐ UK10K ALSPAC Frequency, dbSNP b152 v2	NA000193272.1#23	Variations	
☐ ExAC Release 1 Frequency, dbSNP b152 v2	NA000193272.1#24	Variations	
☐ GnomAD Frequency, dbSNP b152 v2	NA000193272.1#25	Variations	
☐ GO-ESP 6500 Frequency, dbSNP b152 v2	NA000193272.1#26	Variations	
☐ TOPMED Frequency, dbSNP b152 v2	NA000193272.1#27	Variations	
☐ UK10K TWINSUK Frequency, dbSNP b152 v2	NA000193272.1#28	Variations	
☐ Stop Gain or Loss Variations, dbSNP b152 v2	NA000193272.1#29	Variations	
☐ Third Party Annot, dbSNP b152 v2	NA000193272.1#3	Variations	
☐ In-frame Any, dbSNP b152 v2	NA000193272.1#30	Variations	
☐ 3' Gene Region, dbSNP b152 v2	NA000193272.1#4	Variations	
☐ 5' Gene Region, dbSNP b152 v2	NA000193272.1#5	Variations	
☐ 5' UTR Region, dbSNP b152 v2	NA000193272.1#6	Variations	
☐ 3' UTR Region, dbSNP b152 v2	NA000193272.1#7	Variations	
☐ Intron Region, dbSNP b152 v2	NA000193272.1#8	Variations	
☐ Splice Donor Region Variations, dbSNP b152 v2	NA000193272.1#9	Variations	
☐ Cited Variants, dbSNP Build 150 (Homo sapiens Annotation Release 108)	NA000000067.16	Variations	

Fig. 6 Graphical Sequence View—Track Configuration Dialog to choose tracks to display

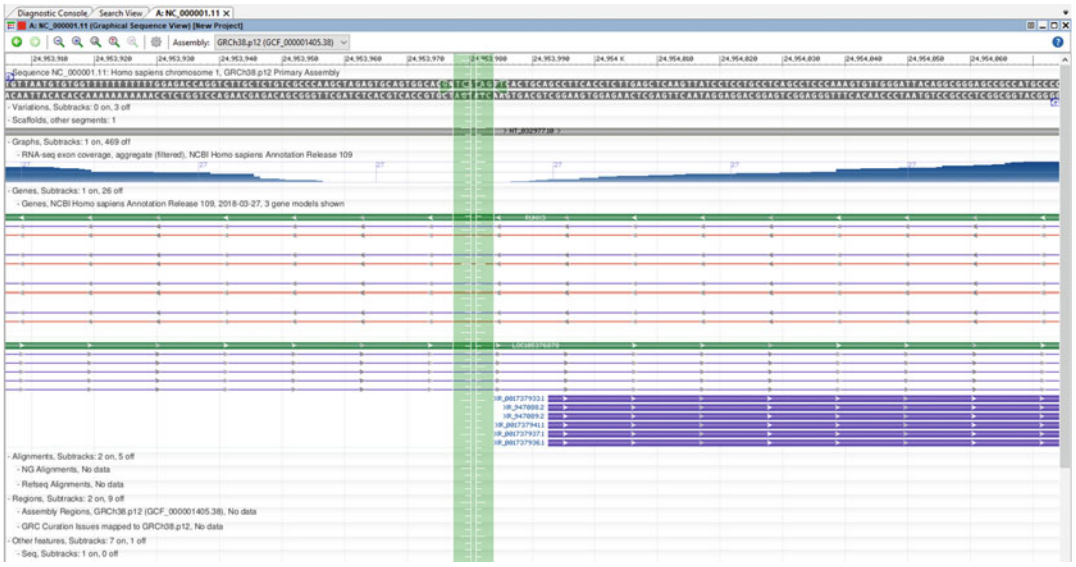
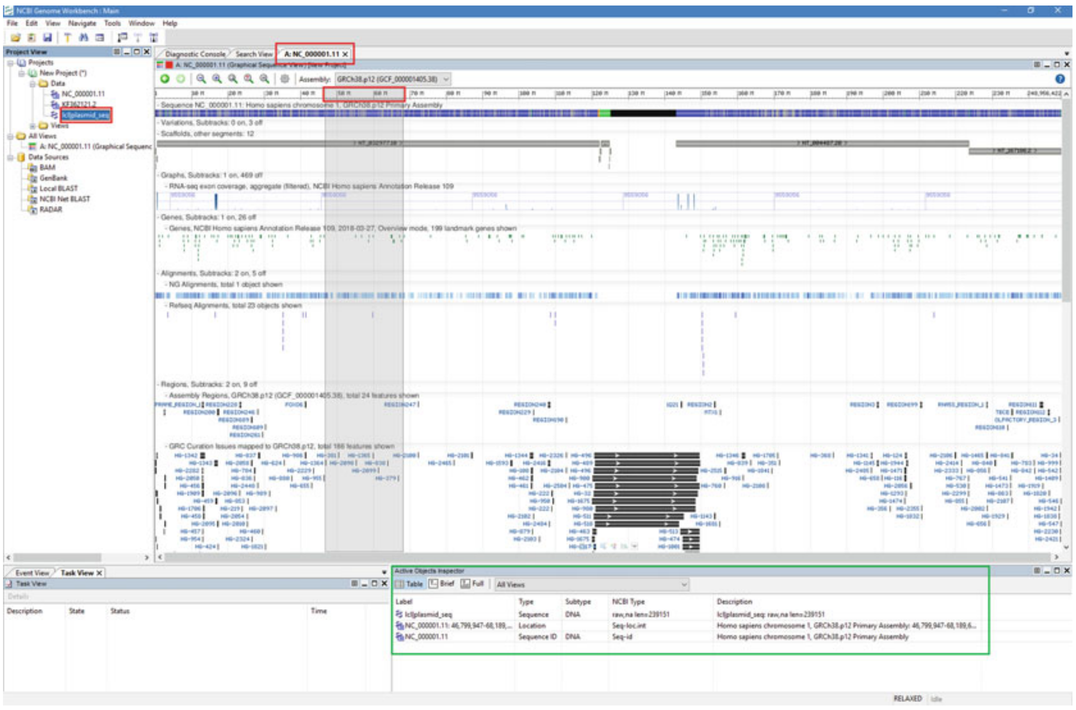

Diagnostic Console / Search View / A: NC_000001.11
A: NC_000001.11 (Graphical Sequence View) [New Project]
Assembly: GRCh38.p12 (GCF_000001405.38)
Sequence NC_000001.11: Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly
Variations, Subtracks: 0 on, 3 off
Scaffolds, other segments: 1
Graphs, Subtracks: 1 on, 469 off
RNA-seq exon coverage, aggregate (filtered), NCBI Homo sapiens Annotation Release 109
Genes, Subtracks: 1 on, 26 off
Genes, NCBI Homo sapiens Annotation Release 109, 2018-03-27, 3 gene models shown
Alignments, Subtracks: 2 on, 5 off
NG Alignments, No data
Refseq Alignments, No data
Regions, Subtracks: 2 on, 9 off
Assembly Regions, GRCh38.p12 (GCF_000001405.38), No data
GRC Curation Issues mapped to GRCh38.p12, No data
Other features, Subtracks: 7 on, 1 off
Seq, Subtracks: 1 on, 0 off

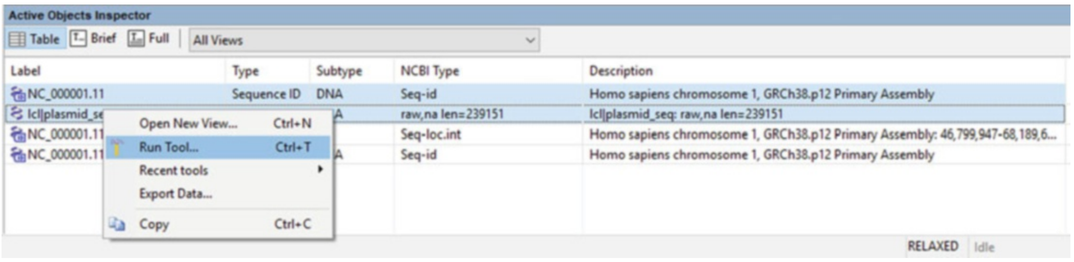
Fig. 7 Graphical Sequence View—zooming into details using zoom ruler (green)

The Active Object Inspector can be used to specify targets for running tools (Fig. 9). Many tools, like NCBI BLAST, can use the selected area to specify the search (query or subject in BLAST terminology).



Label	Type	Subtype	NCBI Type	Description
lcl|plasmid_seq	Sequence	DNA	raw,na len=239151	lcl|plasmid_seq: raw,na len=239151
NC_000001.11: 46,799,947-68,189,...	Location		Seq-loc.int	Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly: 46,799,947-68,189,6...
NC_000001.11	Sequence ID	DNA	Seq-id	Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly

Fig. 8 Active Object Inspector (All Views mode)—shows all selected molecules and locations on the genome



Label	Type	Subtype	NCBI Type	Description
NC_000001.11	Sequence ID	DNA	Seq-id	Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly
lcl|plasmid_se		A	raw,na len=239151	lcl|plasmid_seq: raw,na len=239151
NC_000001.11			Seq-loc.int	Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly: 46,799,947-68,189,6...
NC_000001.11		A	Seq-id	Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly

Fig. 9 Active Object Inspector used to run a tool on two selected molecules

2.4 Running Tools

Here are a few basic ways to run tools in Genome Workbench.

One approach is to select molecules of interest and use the Tools menu to pass selected objects there (see Figs. 9 and 10).

Genome Workbench will attempt to extract the appropriate content from any selection for use in a selected tool. For example, if the user selects not sequence data but annotations, alignments, or any other data type which refers to sequences or locations on a genome, the tool will try to extract those sequences or genomic locations and offer all compatible data as a tool argument candidate.

In the example here, the NCBI BLAST Tool is selected (Fig. 11). Each tool guides the user through a step-by-step wizard interface to help to establish the run options and parameters (Figs. 12 and 13).

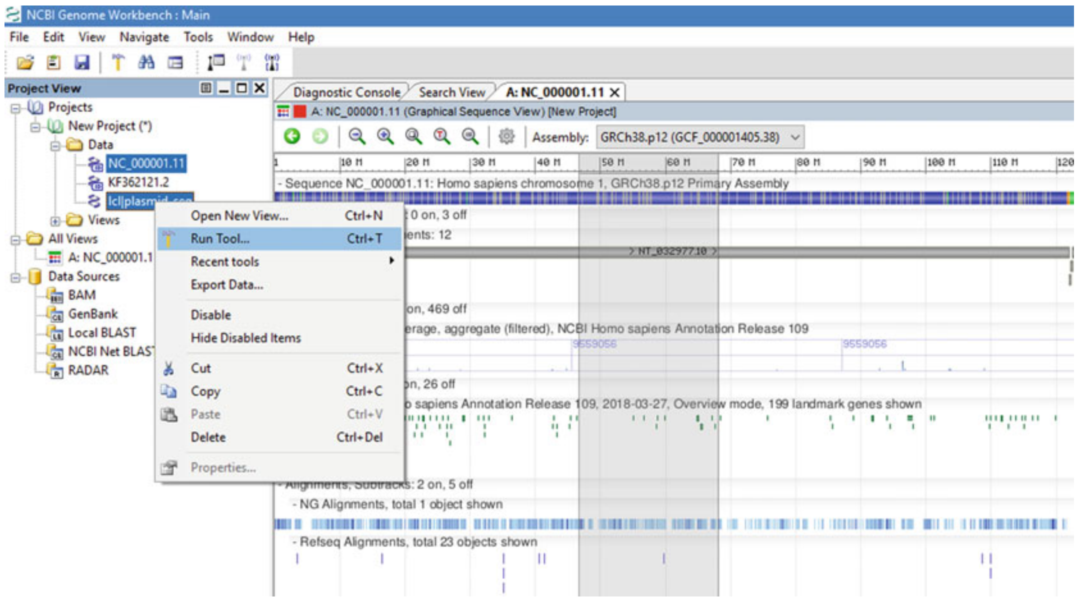


Fig. 10 Project View context menu to Run Tool

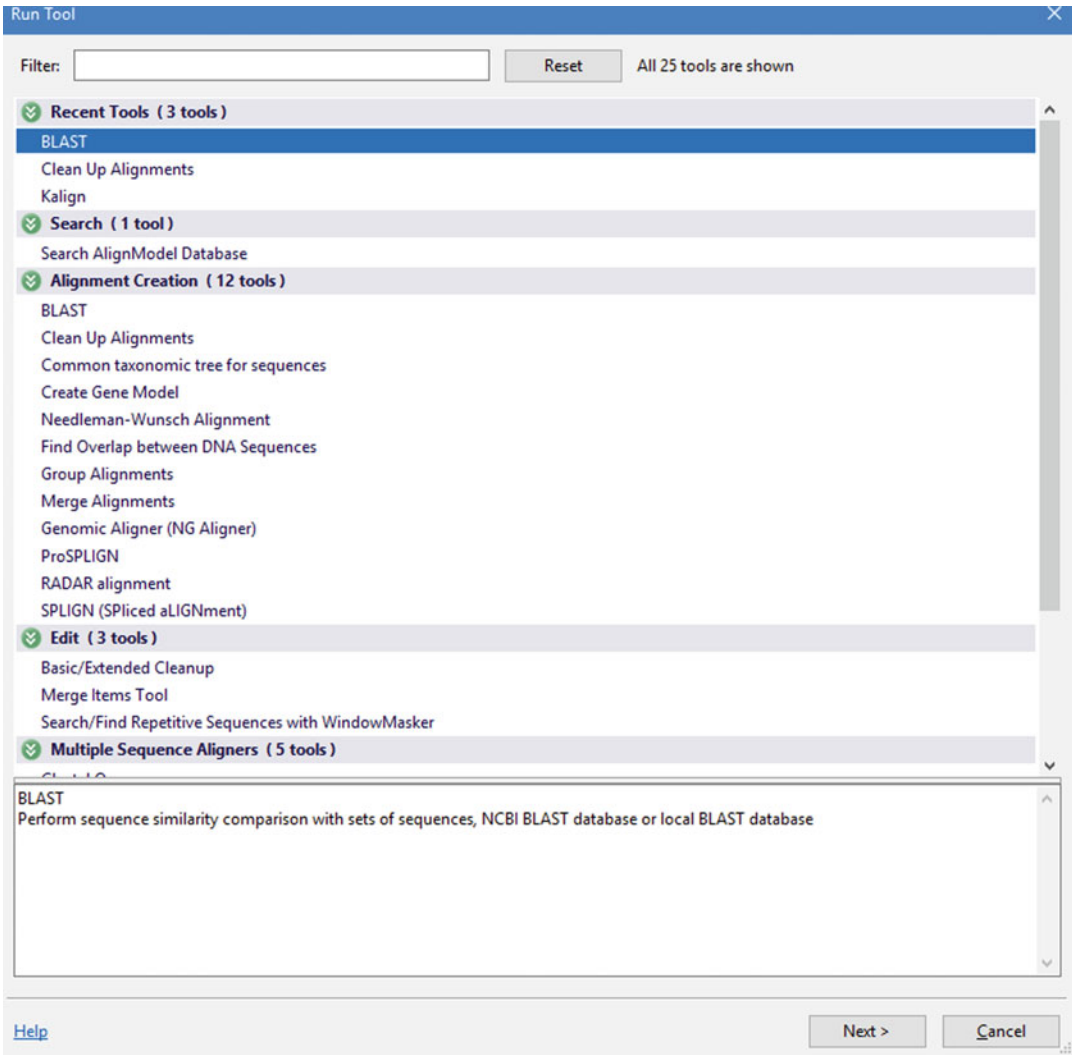
Some tools will run for an extended period of time. The run-time status will be reported in the Task View display. A BLAST run produces an alignment(s) which is automatically loaded into the project and displayed in Graphical Sequence View (Fig. 14).

3 Phylogenetic Analysis Using BLAST Search, Multiple Alignments, and Phylogenetic Tree View

This use case analyzes the phylogenetic relationship between two clades: Xerula/Oudemansiella and Rubescent Amanita spp. The user selected sequence KJ620018 as a query.

1. Load query sequence into project from GenBank.
2. Run BLAST alignment tool using BLASTn, or preferable blast program.
3. Run Multiple Alignment tool on results found by BLAST.
4. Run tool to create phylogenetic tree.

NOTE: KJ620018.1 is sequence from Oudemansiella canarii. It includes a partial internal transcribed spacer 1 (ITS1), a 5.8S rRNA gene, and a partial internal transcribed spacer 2 (ITS2) (https://www.ncbi.nlm.nih.gov/nuccore/KJ620018.1). It is known that in Opisthokonts ITS1 is located between 18S and 5.8S, and ITS2 is located between 5.8S and 28S rRNA genes in a ribosomal cluster.


Run Tool

Filter: Reset All 25 tools are shown

Recent Tools (3 tools)
BLAST
Clean Up Alignments
Kalign
Search (1 tool)
Search AlignModel Database
Alignment Creation (12 tools)
BLAST
Clean Up Alignments
Common taxonomic tree for sequences
Create Gene Model
Needleman-Wunsch Alignment
Find Overlap between DNA Sequences
Group Alignments
Merge Alignments
Genomic Aligner (NG Aligner)
ProSPLIGN
RADAR alignment
SPLIGN (SPliced aLIGNment)
Edit (3 tools)
Basic/Extended Cleanup
Merge Items Tool
Search/Find Repetitive Sequences with WindowMasker
Multiple Sequence Aligners (5 tools)

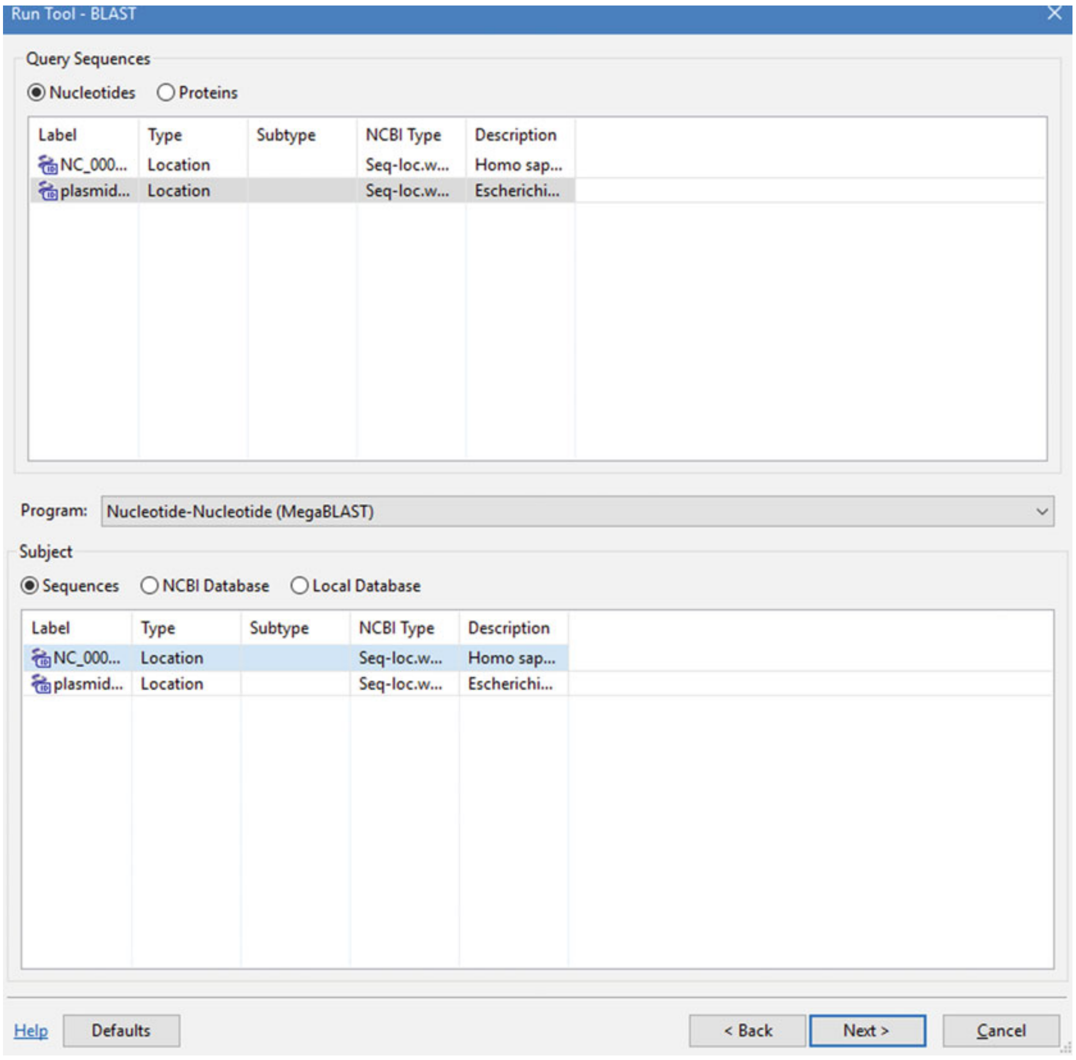
BLAST
Perform sequence similarity comparison with sets of sequences, NCBI BLAST database or local BLAST database

Help Next > Cancel

Fig. 11 Run Tool dialog. BLAST is selected

Click on the Tool icon to open the **Run Tool** dialog (shortcut: Ctrl+T, Fig. 15). In the **Run Tool** Dialog/**Alignment Creation** section, select BLAST and click the **Next** button to open the **Run Tool BLAST** dialog (Fig. 16). In this dialog, select Nucleotide as the query sequence type (if not already selected), select the BLASTn program to run (from drop-down list), and select NCBI Database/nt Nucleotide collection as the Subject. Please note, that in this case, user does not run pairwise alignment but uses BLAST to run a search against a database hosted at the NCBI.

Click **Next** to see general parameters and more options to restrict BLAST search (Fig. 17). Make sure that option "Link related hits together" is NOT checked. The current example will use default parameters.



Query Sequences: (●) Nucleotides ( ) Proteins

Label	Type	Subtype	NCBI Type	Description
NC_000...	Location		Seq-loc.w...	Homo sap...
plasmid...	Location		Seq-loc.w...	Escherichi...

Program: Nucleotide-Nucleotide (MegaBLAST)

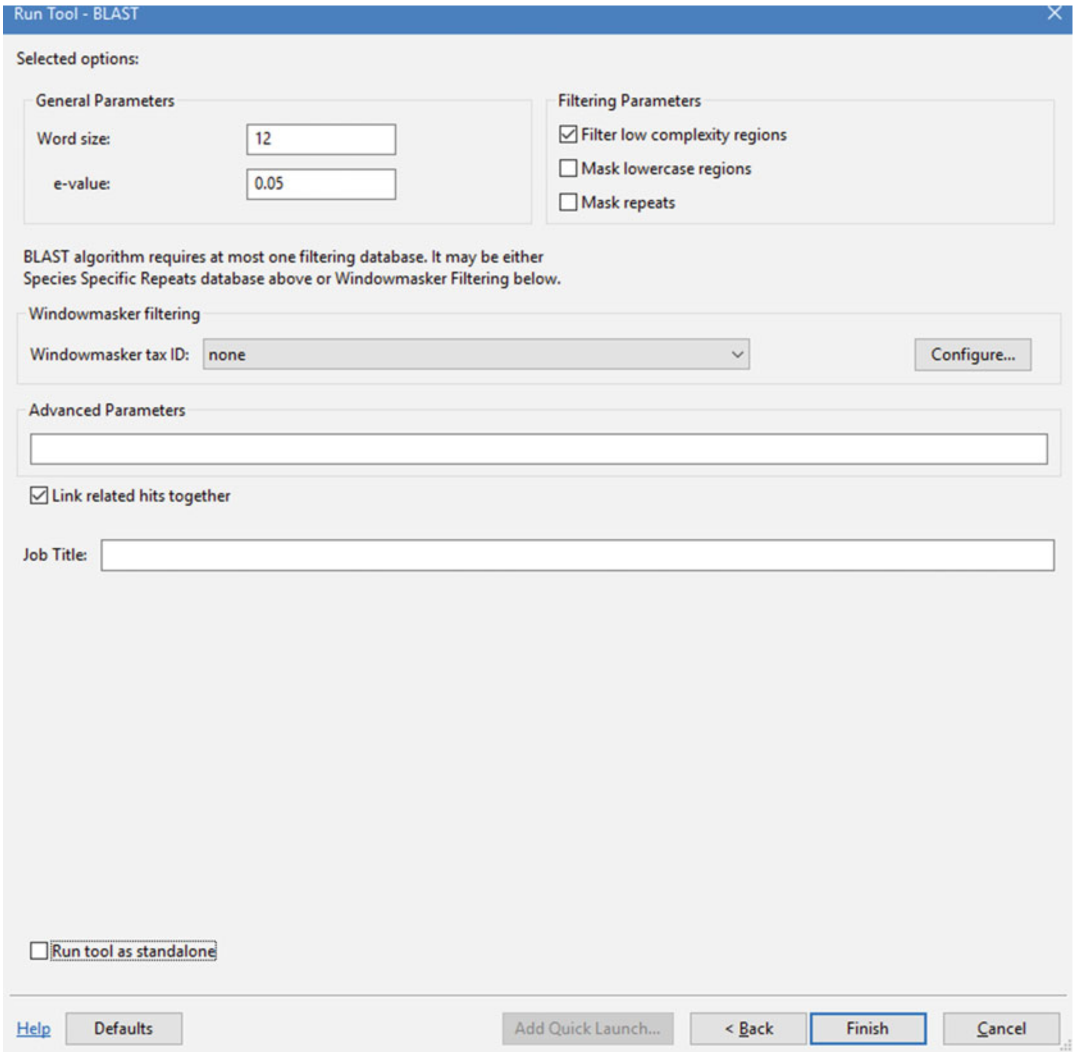
Subject: (●) Sequences ( ) NCBI Database ( ) Local Database

Label	Type	Subtype	NCBI Type	Description
NC_000...	Location		Seq-loc.w...	Homo sap...
plasmid...	Location		Seq-loc.w...	Escherichi...

Fig. 12 Run Tool—use Next button to define tool specific parameters

The Task View window shows the BLAST search status. When the search is completed, the result will be automatically loaded in the Project View. The result can be open in different Views, including Alignment Summary View and Multiple Alignment View (Fig. 18). These Views allow the user to see accessions/organisms found by BLAST search.

Select the BLAST result in the Project View and use the context menu to run **Tools**. In the **Tools** dialog, find the **Multiple Sequence Aligners** section, select alignment program, pick **Clustal Omega,** and click the **Next** button. The Clustal Omega Tool dialog will open with all unique accessions returned by the BLASTn search, which in this example will be 501 sequences, but the number will depend on word size and other parameters (Fig. 19). For


Run Tool - BLAST

Selected options:

General Parameters

Word size: 12

e-value: 0.05

Filtering Parameters

☑ Filter low complexity regions

☐ Mask lowercase regions

☐ Mask repeats

BLAST algorithm requires at most one filtering database. It may be either Species Specific Repeats database above or Windowmasker Filtering below.

Windowmasker filtering

Windowmasker tax ID: none

Advanced Parameters

☑ Link related hits together

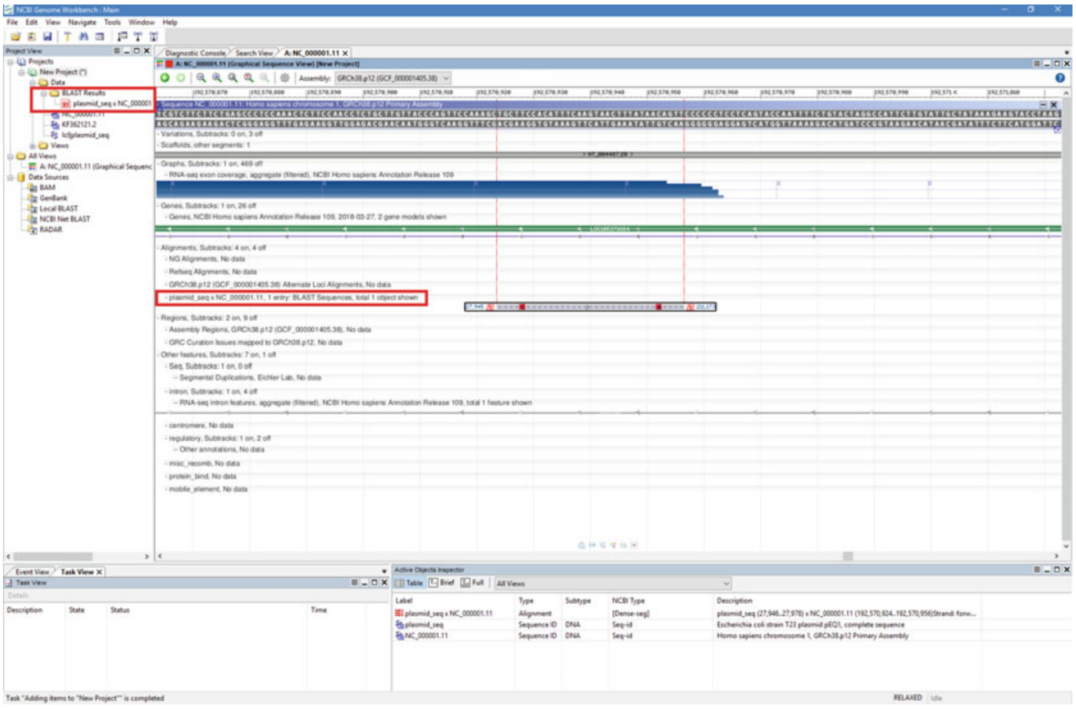
Job Title:

☐ Run tool as standalone

Fig. 13 Run Tool. Sequence BLAST parameters

every accession, the total range of the part of sequence that has hits to the query sequence (KJ620018) is shown. Only this part will be used for the multiple alignment creation. To run multiple aligner, **provide the path to the program** (note: to use the external programs user needs to download, install, and provide the path to the executable) and click the **Finish** button (Fig. 19). It is possible to unselect some sequences and not include them in the multiple alignments.

BLAST in this case used as an initial search stage, returning results as pairwise alignments between the query and the BLAST database sequences, but does not provide alignments between database sequences. Multiple aligners are capable of building a global comparative alignment model on sequences returned by


NCBI Genome Workbench : Main
File Edit View Navigate Tools Window Help
Project View
Projects
New Project (*)
Data
BLAST Results
plasmid_seq x NC_000001
NC_000001.11
KF362121.2
lcl|plasmid_seq
Views
All Views
A: NC_000001.11 (Graphical Sequence)
Data Sources
BAM
GenBank
Local BLAST
NCBI Net BLAST
RADAR
Diagnostic Console
Search View
A: NC_000001.11
Assembly: GRCh38.p12 (GCF_000001405.38)
Sequence NC_000001.11: Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly
Variations, Subtracks: 0 on, 3 off
Scaffolds, other segments: 1
Graphs, Subtracks: 1 on, 469 off
RNA-seq exon coverage, aggregate (filtered), NCBI Homo sapiens Annotation Release 109
Genes, Subtracks: 1 on, 26 off
Genes, NCBI Homo sapiens Annotation Release 109, 2018-03-27, 2 gene models shown
Alignments, Subtracks: 4 on, 4 off
NG Alignments, No data
Refseq Alignments, No data
GRCh38.p12 (GCF_000001405.38) Alternate Loci Alignments, No data
plasmid_seq x NC_000001.11, 1 entry: BLAST Sequences, total 1 object shown
Regions, Subtracks: 2 on, 9 off
Assembly Regions, GRCh38.p12 (GCF_000001405.38), No data
GRC Curation Issues mapped to GRCh38.p12, No data
Other features, Subtracks: 7 on, 1 off
Seq, Subtracks: 1 on, 0 off
Segmental Duplications, Eichler Lab, No data
Intron, Subtracks: 1 on, 4 off
RNA-seq intron features, aggregate (filtered), NCBI Homo sapiens Annotation Release 109, total 1 feature shown
centromere, No data
regulatory, Subtracks: 1 on, 2 off
Other annotations, No data
misc_recomb, No data
protein_bind, No data
mobile_element, No data
Event View
Task View
Description
State
Status
Time
Active Objects Inspector
Table
Brief
Full
All Views
Task "Adding items to "New Project"" is completed
RELAXED
Idle

Label	Type	Subtype	NCBI Type	Description
plasmid_seq x NC_000001.11	Alignment		[Dense-seg]	plasmid_seq (27,946..27,978) x NC_000001.11 (192,570,924..192,570,956)(Strand: forw...
plasmid_seq	Sequence ID	DNA	Seq-id	Escherichia coli strain T23 plasmid pEQ1, complete sequence
NC_000001.11	Sequence ID	DNA	Seq-id	Homo sapiens chromosome 1, GRCh38.p12 Primary Assembly

Fig. 14 Project View with loaded BLAST alignment, alignment is also displayed as a track in Graphical Sequence View

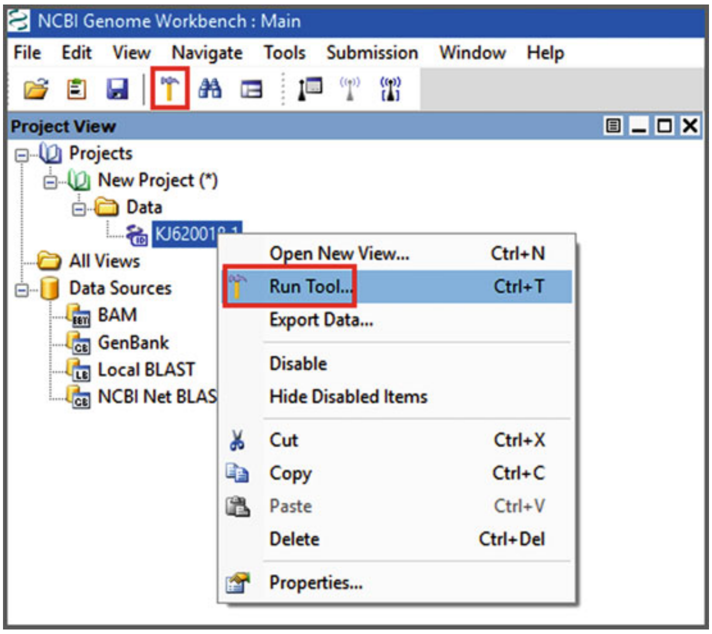
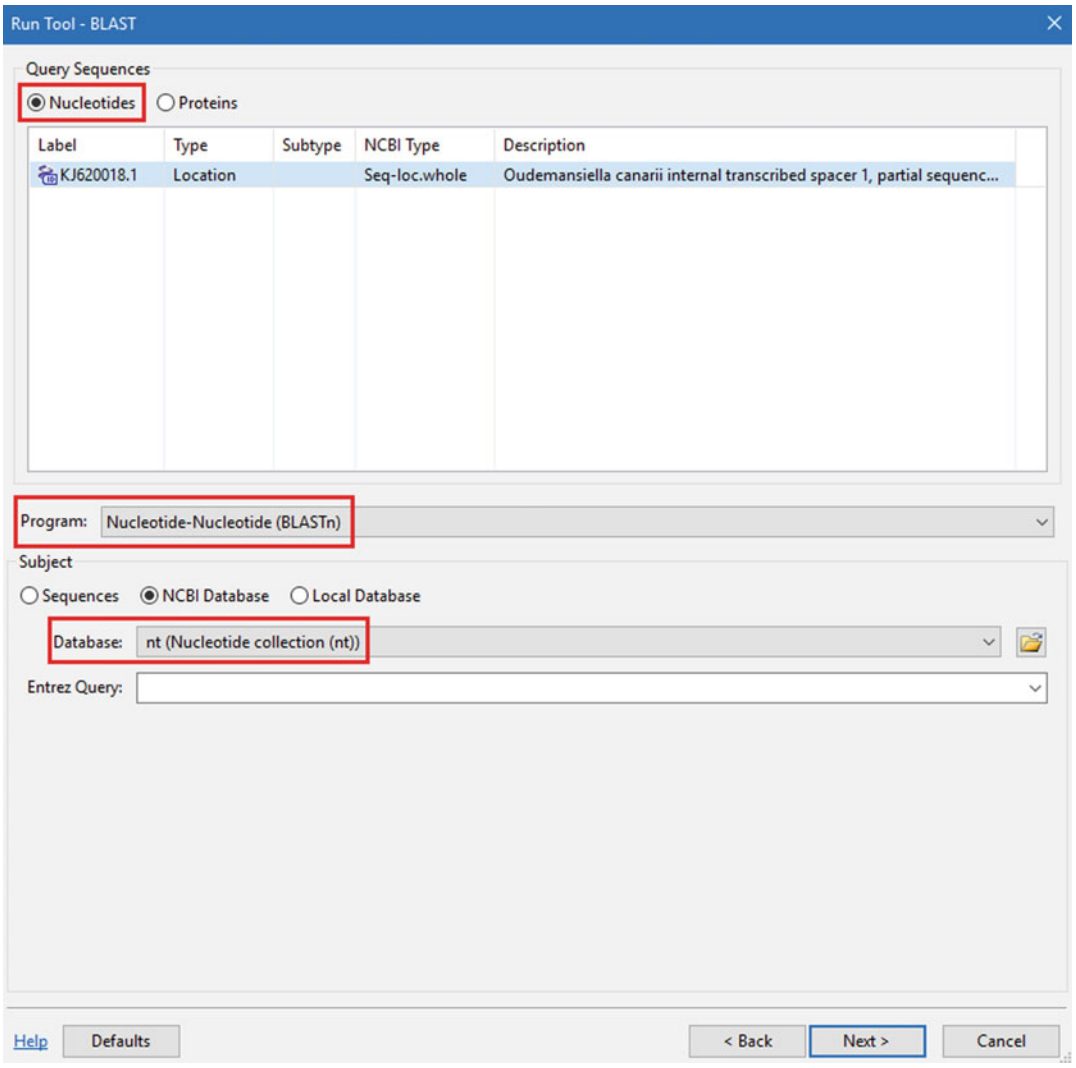


Fig. 15 KJ620018.1 accession loaded Genome Workbench project, ready to Run Blast


Run Tool - BLAST

Query Sequences

● Nucleotides ○ Proteins

Label	Type	Subtype	NCBI Type	Description
KJ620018.1	Location		Seq-loc.whole	Oudemansiella canarii internal transcribed spacer 1, partial sequenc...

Program: Nucleotide-Nucleotide (BLASTn)

Subject

○ Sequences ● NCBI Database ○ Local Database

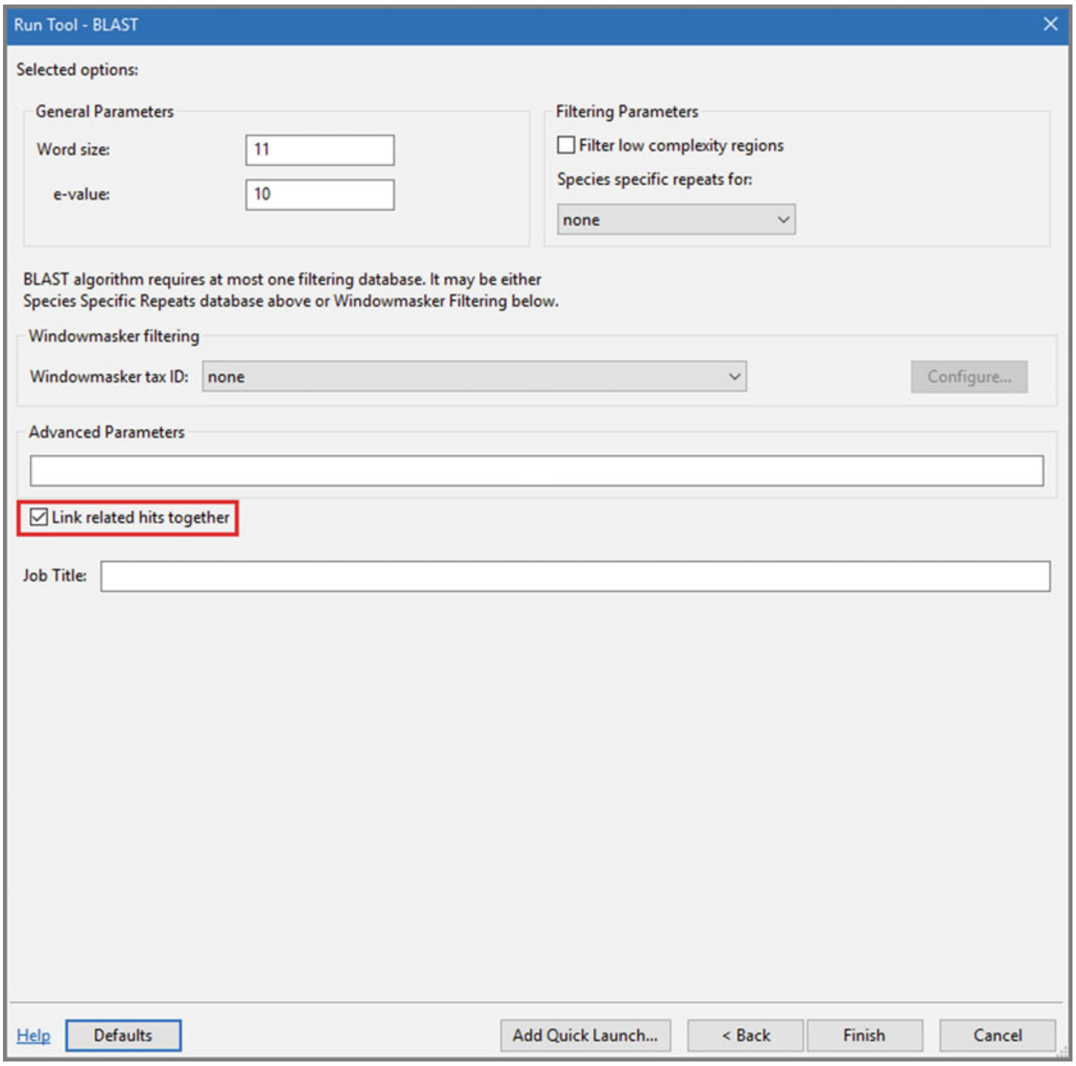
Database: nt (Nucleotide collection (nt))

Entrez Query:

Fig. 16 Tool Blast dialog is used to run search against NT database at NCBI

BLAST and selected by the user as topics of interest. BLAST stage and interactive preselection and pruning greatly improve chances that multiple aligners will be able to produce a quality model.

Some multiple aligners have the option to return the guide tree used for alignment (CLUSTAL) or generate a reconstructed tree from the alignment (MUSCLE). These trees are used for alignment creation, thus they are not truly phylogenetic trees although they are mathematically related. In these trees, sequence IDs are used for labels on leaf nodes (but not organism names, or sequence title). Other tree builders may decorate tree nodes with different attributes. The Tree View allows the user to configure custom node labels to use non-default attributes.


Run Tool - BLAST

Selected options:

General Parameters

Word size: 11

e-value: 10

Filtering Parameters

☐ Filter low complexity regions

Species specific repeats for: none

BLAST algorithm requires at most one filtering database. It may be either Species Specific Repeats database above or Windowmasker Filtering below.

Windowmasker filtering

Windowmasker tax ID: none

Advanced Parameters

☑ Link related hits together

Job Title:

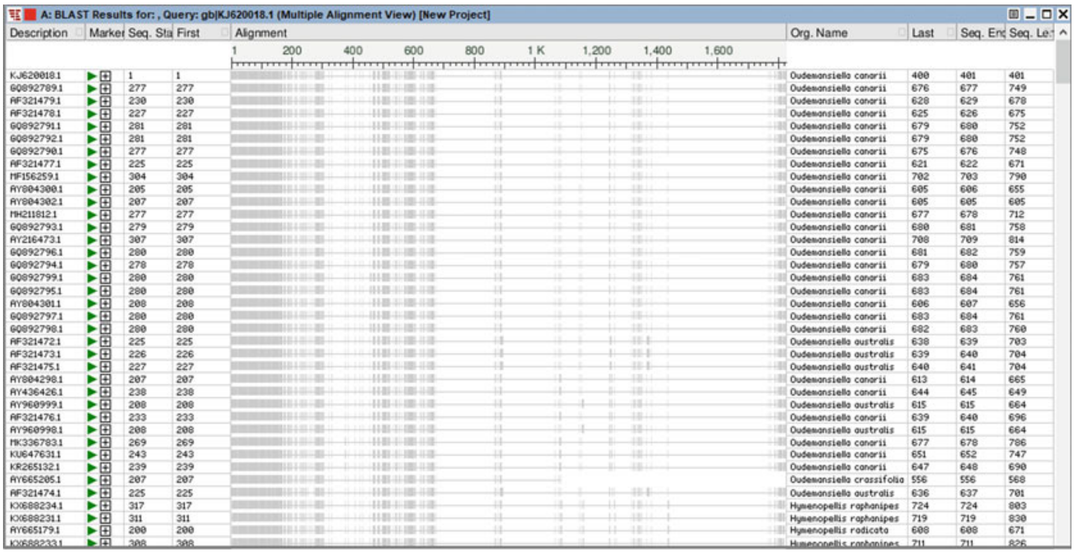
Fig. 17 Specify BLAST parameters like word size

Wait for multiple alignment creation. The result will load and appear in the Project View. Multiple Sequence Alignment Viewer can be opened to inspect the alignment (Fig. 20).

Now the user can run another tool: Phylogenetic Tree Builder Tool, click Next and select Distance Method: Jukes-Cantor (DNA), Tree Construct method: Neighbor Joining, and Labels for Leaf Nodes: Taxonomic Name (Fig. 21).

Note: it is also possible to create tree with "Labels for Leaf Nodes: Sequence ID" (default option). Phylogenetic Tree Builder tool in Genome Workbench is based on phylogeny reconstruction algorithms described in [7, 8].

Visualization of phylogenetic tree can be done using Tree View (see Fig. 22).



A: BLAST Results for: , Query: gb|KJ620018.1 (Multiple Alignment View) [New Project]

Description	Marker	Seq. Sta	First	Alignment	Org. Name	Last	Seq. End	Seq. Le
KJ620018.1	▶ ⊞	1	1		Oudemansiella canarii	400	401	401
GQ892789.1	▶ ⊞	277	277		Oudemansiella canarii	676	677	749
AF321479.1	▶ ⊞	230	230		Oudemansiella canarii	628	629	678
AF321478.1	▶ ⊞	227	227		Oudemansiella canarii	625	626	675
GQ892791.1	▶ ⊞	281	281		Oudemansiella canarii	679	680	752
GQ892792.1	▶ ⊞	281	281		Oudemansiella canarii	679	680	752
GQ892790.1	▶ ⊞	277	277		Oudemansiella canarii	675	676	748
AF321477.1	▶ ⊞	225	225		Oudemansiella canarii	621	622	671
MF156259.1	▶ ⊞	304	304		Oudemansiella canarii	702	703	790
AY804300.1	▶ ⊞	205	205		Oudemansiella canarii	605	606	655
AY804302.1	▶ ⊞	207	207		Oudemansiella canarii	605	605	605
MH211812.1	▶ ⊞	277	277		Oudemansiella canarii	677	678	712
GQ892793.1	▶ ⊞	279	279		Oudemansiella canarii	680	681	758
AY216473.1	▶ ⊞	307	307		Oudemansiella canarii	708	709	814
GQ892796.1	▶ ⊞	280	280		Oudemansiella canarii	681	682	759
GQ892794.1	▶ ⊞	278	278		Oudemansiella canarii	679	680	757
GQ892799.1	▶ ⊞	280	280		Oudemansiella canarii	683	684	761
GQ892795.1	▶ ⊞	280	280		Oudemansiella canarii	683	684	761
AY804301.1	▶ ⊞	208	208		Oudemansiella canarii	606	607	656
GQ892797.1	▶ ⊞	280	280		Oudemansiella canarii	683	684	761
GQ892798.1	▶ ⊞	280	280		Oudemansiella canarii	682	683	760
AF321472.1	▶ ⊞	225	225		Oudemansiella australis	638	639	703
AF321473.1	▶ ⊞	226	226		Oudemansiella australis	639	640	704
AF321475.1	▶ ⊞	227	227		Oudemansiella australis	640	641	704
AY804298.1	▶ ⊞	207	207		Oudemansiella canarii	613	614	665
AY436426.1	▶ ⊞	238	238		Oudemansiella canarii	644	645	649
AY960999.1	▶ ⊞	208	208		Oudemansiella australis	615	615	664
AF321476.1	▶ ⊞	233	233		Oudemansiella canarii	639	640	696
AY960998.1	▶ ⊞	208	208		Oudemansiella australis	615	615	664
MK336783.1	▶ ⊞	269	269		Oudemansiella canarii	677	678	786
KU647631.1	▶ ⊞	243	243		Oudemansiella canarii	651	652	747
KR265132.1	▶ ⊞	239	239		Oudemansiella canarii	647	648	690
AY665205.1	▶ ⊞	207	207		Oudemansiella crassifolia	556	556	568
AF321474.1	▶ ⊞	225	225		Oudemansiella australis	636	637	701
KX688234.1	▶ ⊞	317	317		Hymenopellis raphanipes	724	724	803
KX688231.1	▶ ⊞	311	311		Hymenopellis raphanipes	719	719	830
AY665179.1	▶ ⊞	200	200		Hymenopellis radicata	608	608	671
KX688233.1	▶ ⊞	308	308		Hymenopellis raphanipes	711	711	826

Fig. 18 Genome Workbench Multiple Alignment View shows BLAST search results

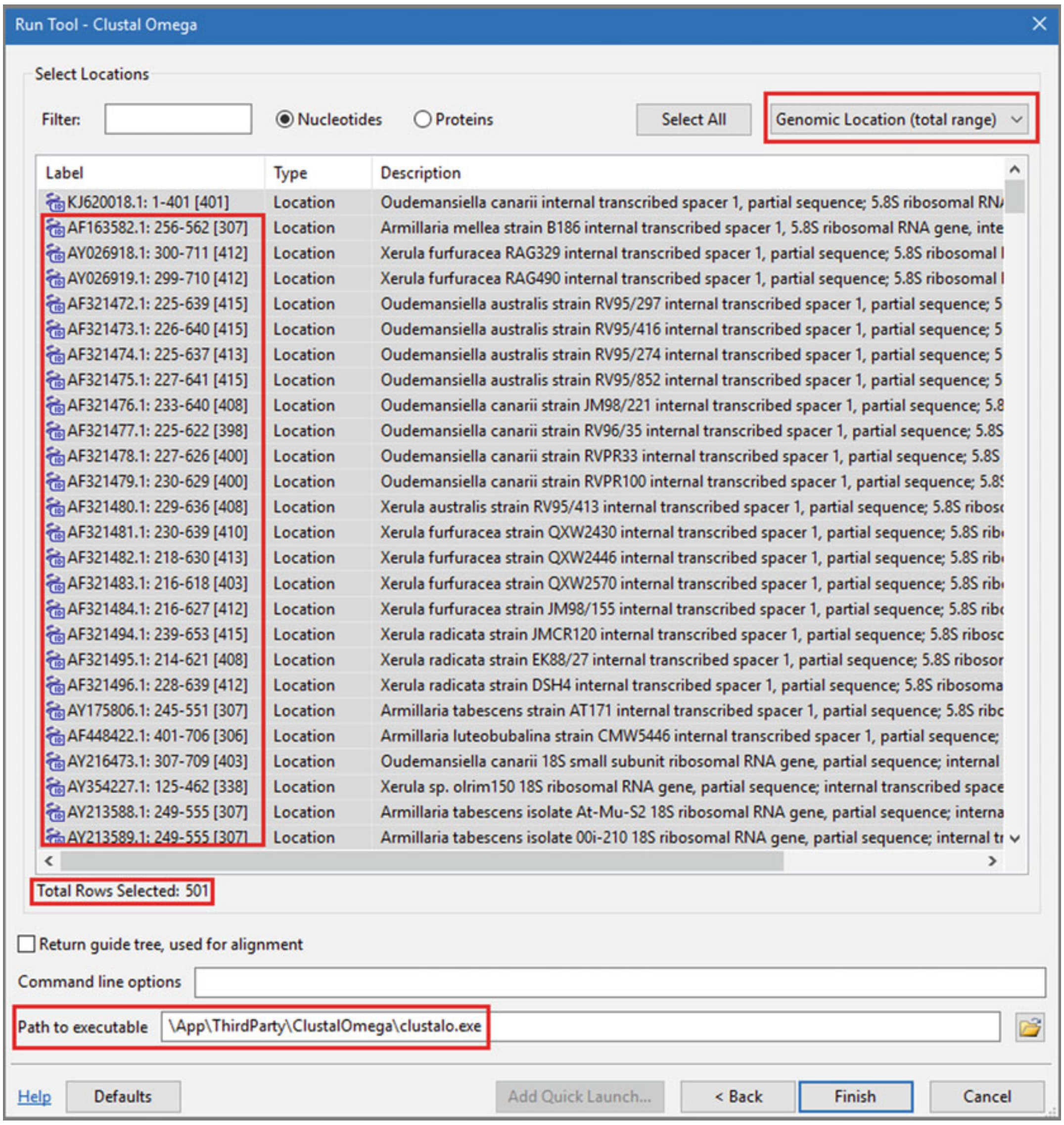
Initial raw topology and navigation can be improved with right click context menu, use **Zoom behavior**- > **Vertical** and select **Re-Root**- > **Set Midpoint Root** (Fig. 23**).**

Tree View supports fully interactive zoom in and out so that the user can see labels. If the user prefers to see topology only (without distances), use right click menu/**Layout** and remove the checkmark from the **Use Distances** option, or use **Layout/Slanted Cladogram**.

If a user has obtained genome sequence data using a sequencing technology that produces results very quickly, but with low quality, the user might want to create an alignment with sequences from GenBank for the same or very similar organisms and identify regions with very low identity as candidates for resequencing as part of the finishing process.

As illustrated previously, the BLAST Tool can be used to compare a sequence with the NT database to assist in identifying the organism from which the sequence was created to ensure accurate taxonomic assignment [9]. Multiple sequence alignments could also be used to align pathogen sequences with non-toxic forms of the same pathogen to identify genes that are responsible for pathogenicity and virulence, or to identify variations in extremophile organisms that might be responsible for their ability to exist in those environments.

When these analyses are complete, the user may decide that the sequence has scientific value and should be submitted to GenBank. Genome Workbench v.3.0.0 offers the Editing Package as a suite of tools for submission preparation.



Genomic Location (total range)

(●) Nucleotides ( ) Proteins

Label	Type	Description
KJ620018.1: 1-401 [401]	Location	Oudemansiella canarii internal transcribed spacer 1, partial sequence; 5.8S ribosomal RN/
AF163582.1: 256-562 [307]	Location	Armillaria mellea strain B186 internal transcribed spacer 1, 5.8S ribosomal RNA gene, inte
AY026918.1: 300-711 [412]	Location	Xerula furfuracea RAG329 internal transcribed spacer 1, partial sequence; 5.8S ribosomal I
AY026919.1: 299-710 [412]	Location	Xerula furfuracea RAG490 internal transcribed spacer 1, partial sequence; 5.8S ribosomal I
AF321472.1: 225-639 [415]	Location	Oudemansiella australis strain RV95/297 internal transcribed spacer 1, partial sequence; 5
AF321473.1: 226-640 [415]	Location	Oudemansiella australis strain RV95/416 internal transcribed spacer 1, partial sequence; 5
AF321474.1: 225-637 [413]	Location	Oudemansiella australis strain RV95/274 internal transcribed spacer 1, partial sequence; 5
AF321475.1: 227-641 [415]	Location	Oudemansiella australis strain RV95/852 internal transcribed spacer 1, partial sequence; 5
AF321476.1: 233-640 [408]	Location	Oudemansiella canarii strain JM98/221 internal transcribed spacer 1, partial sequence; 5.8
AF321477.1: 225-622 [398]	Location	Oudemansiella canarii strain RV96/35 internal transcribed spacer 1, partial sequence; 5.8S
AF321478.1: 227-626 [400]	Location	Oudemansiella canarii strain RVPR33 internal transcribed spacer 1, partial sequence; 5.8S
AF321479.1: 230-629 [400]	Location	Oudemansiella canarii strain RVPR100 internal transcribed spacer 1, partial sequence; 5.8S
AF321480.1: 229-636 [408]	Location	Xerula australis strain RV95/413 internal transcribed spacer 1, partial sequence; 5.8S ribos
AF321481.1: 230-639 [410]	Location	Xerula furfuracea strain QXW2430 internal transcribed spacer 1, partial sequence; 5.8S rib
AF321482.1: 218-630 [413]	Location	Xerula furfuracea strain QXW2446 internal transcribed spacer 1, partial sequence; 5.8S rib
AF321483.1: 216-618 [403]	Location	Xerula furfuracea strain QXW2570 internal transcribed spacer 1, partial sequence; 5.8S rib
AF321484.1: 216-627 [412]	Location	Xerula furfuracea strain JM98/155 internal transcribed spacer 1, partial sequence; 5.8S rib
AF321494.1: 239-653 [415]	Location	Xerula radicata strain JMCR120 internal transcribed spacer 1, partial sequence; 5.8S ribosc
AF321495.1: 214-621 [408]	Location	Xerula radicata strain EK88/27 internal transcribed spacer 1, partial sequence; 5.8S ribosor
AF321496.1: 228-639 [412]	Location	Xerula radicata strain DSH4 internal transcribed spacer 1, partial sequence; 5.8S ribosoma
AY175806.1: 245-551 [307]	Location	Armillaria tabescens strain AT171 internal transcribed spacer 1, partial sequence; 5.8S ribc
AF448422.1: 401-706 [306]	Location	Armillaria luteobubalina strain CMW5446 internal transcribed spacer 1, partial sequence;
AY216473.1: 307-709 [403]	Location	Oudemansiella canarii 18S small subunit ribosomal RNA gene, partial sequence; internal
AY354227.1: 125-462 [338]	Location	Xerula sp. olrim150 18S ribosomal RNA gene, partial sequence; internal transcribed space
AY213588.1: 249-555 [307]	Location	Armillaria tabescens isolate At-Mu-S2 18S ribosomal RNA gene, partial sequence; interna
AY213589.1: 249-555 [307]	Location	Armillaria tabescens isolate 00i-210 18S ribosomal RNA gene, partial sequence; internal tr

Total Rows Selected: 501

[ ] Return guide tree, used for alignment

Command line options

Path to executable \App\ThirdParty\ClustalOmega\clustalo.exe

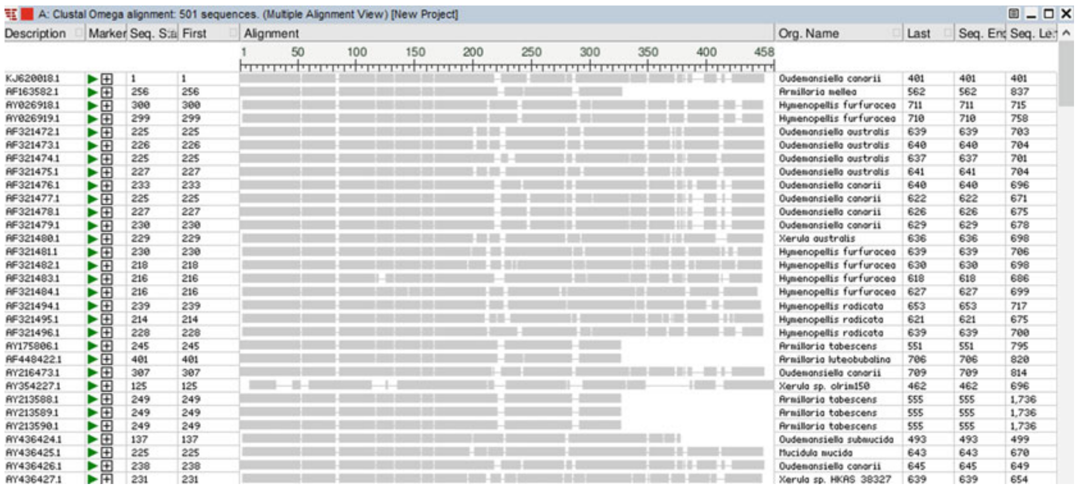
Fig. 19 Run Tool—Clustal Omega on BLAST found sequences

4 Genome Workbench Submission Preparation

4.1 Getting Started: Enable Editing Package

Genome Workbench v.3.0.0 replaces NCBI Sequin as a new generation of tools for GenBank [5] submission preparation and editing. In order to use Genome Workbench to prepare a submission for GenBank, the Editing Package needs to be enabled. To do this, use the Tools menu and select Packages (Fig. 24).

Then locate and select the Editing Package (Fig. 25). The Package has already been enabled if the dialog shows it as "Loaded." Otherwise check the Enable box, click OK, and restart Genome Workbench.



A: Clustal Omega alignment: 501 sequences. (Multiple Alignment View) [New Project]

Description	Marker	Seq. S	First	Alignment	Org. Name	Last	Seq. En	Seq. Le
KJ620018.1	▶ ⊞	1	1		Oudemansiella canarii	401	401	401
AF163582.1	▶ ⊞	256	256		Armillaria mellea	562	562	837
AY026918.1	▶ ⊞	300	300		Hymenopellis furfuracea	711	711	715
AY026919.1	▶ ⊞	299	299		Hymenopellis furfuracea	710	710	758
AF321472.1	▶ ⊞	225	225		Oudemansiella australis	639	639	703
AF321473.1	▶ ⊞	226	226		Oudemansiella australis	640	640	704
AF321474.1	▶ ⊞	225	225		Oudemansiella australis	637	637	701
AF321475.1	▶ ⊞	227	227		Oudemansiella australis	641	641	704
AF321476.1	▶ ⊞	233	233		Oudemansiella canarii	640	640	696
AF321477.1	▶ ⊞	225	225		Oudemansiella canarii	622	622	671
AF321478.1	▶ ⊞	227	227		Oudemansiella canarii	626	626	675
AF321479.1	▶ ⊞	230	230		Oudemansiella canarii	629	629	678
AF321480.1	▶ ⊞	229	229		Xerula australis	636	636	698
AF321481.1	▶ ⊞	230	230		Hymenopellis furfuracea	639	639	706
AF321482.1	▶ ⊞	218	218		Hymenopellis furfuracea	630	630	698
AF321483.1	▶ ⊞	216	216		Hymenopellis furfuracea	618	618	686
AF321484.1	▶ ⊞	216	216		Hymenopellis furfuracea	627	627	699
AF321494.1	▶ ⊞	239	239		Hymenopellis radicata	653	653	717
AF321495.1	▶ ⊞	214	214		Hymenopellis radicata	621	621	675
AF321496.1	▶ ⊞	228	228		Hymenopellis radicata	639	639	700
AY175806.1	▶ ⊞	245	245		Armillaria tabescens	551	551	795
AF448422.1	▶ ⊞	401	401		Armillaria luteobubalina	706	706	820
AY216473.1	▶ ⊞	307	307		Oudemansiella canarii	709	709	814
AY354227.1	▶ ⊞	125	125		Xerula sp. olrim150	462	462	696
AY213588.1	▶ ⊞	249	249		Armillaria tabescens	555	555	1,736
AY213589.1	▶ ⊞	249	249		Armillaria tabescens	555	555	1,736
AY213590.1	▶ ⊞	249	249		Armillaria tabescens	555	555	1,736
AY436424.1	▶ ⊞	137	137		Oudemansiella submucida	493	493	499
AY436425.1	▶ ⊞	225	225		Mucidula mucida	643	643	670
AY436426.1	▶ ⊞	238	238		Oudemansiella canarii	645	645	649
AY436427.1	▶ ⊞	231	231		Xerula sp. HKAS 38327	639	639	654

Fig. 20 Multiple Alignment View shows results from Clustal Omega

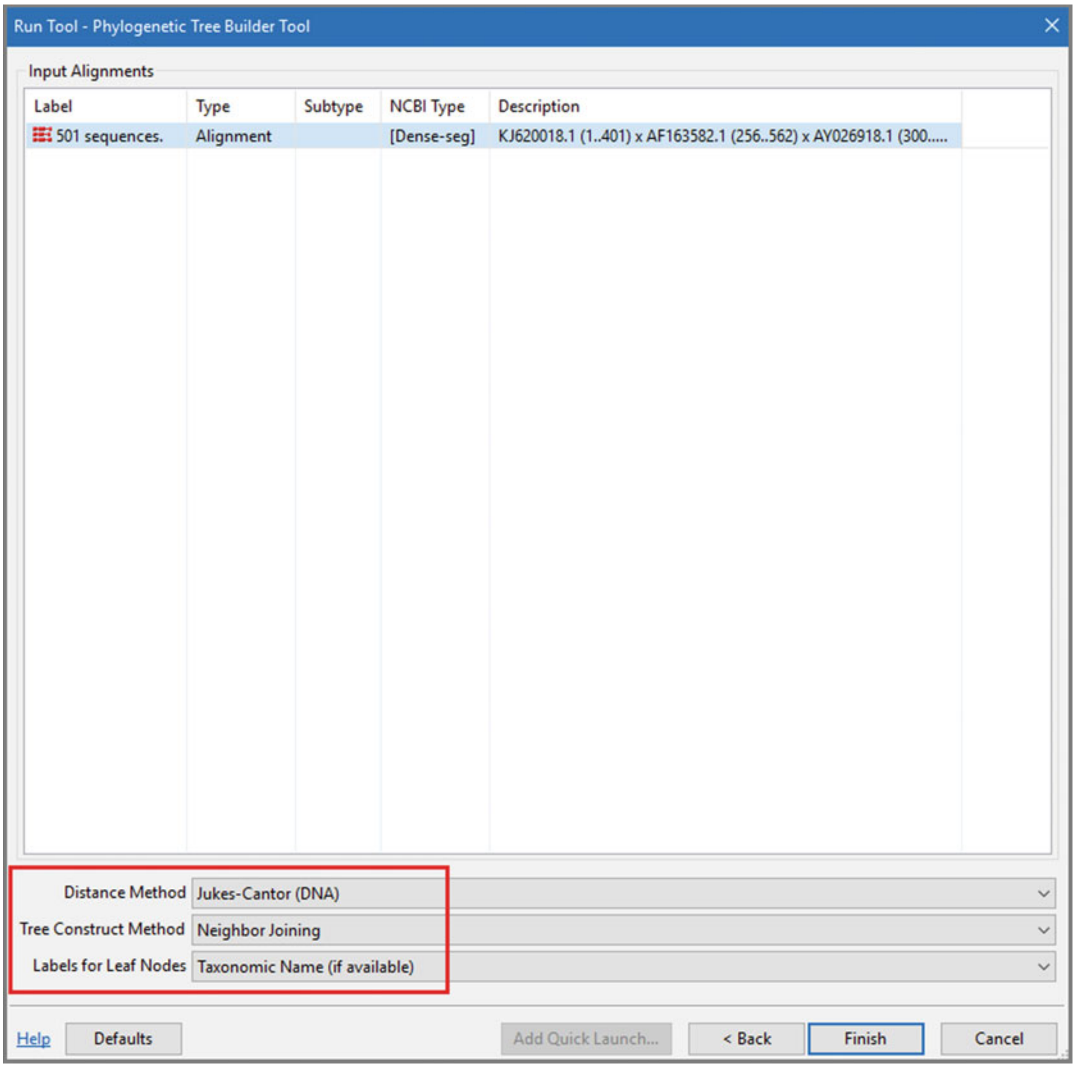
Once Genome Workbench has restarted, a new menu item will appear called "Submission" (Fig. 26).

4.2 Submission Preparation Workflow

Once the Genome Workbench Editing Package is activated, one can use a workflow procedure to load, validate, edit, and submit your data to NCBI. The editing workflow diagram describes the steps to prepare a submission (Fig. 27).

4.3 Submission Process Explained

1. **Use Genome Submission Wizard to prepare for submission to GenBank**
   The Genome Submission Wizard is the first item in the Submission menu. It is used to provide information about the sequence data that is needed for submission to GenBank. This information includes:
   - Contact information to be used by GenBank indexers while the submission is being processed.
   - The identity of the author of the sequences so that other scientists can acknowledge the author in publications that reference the data or contact the author with questions.
   - Publications that refer to the sequence data. GenBank and PubMed provide reciprocal links between publications and supporting data where possible, to help drive new discoveries. Sequence data can be held until publications are published but must be released after that.
   - The biological source of the material that was sequenced. Genome Workbench provides mechanisms to fill in commonly known information about organisms (such as lineage, genetic code, etc.) given taxonomic names. Genome Workbench also allows the user to provide links to other databases


Run Tool - Phylogenetic Tree Builder Tool

Input Alignments

Label	Type	Subtype	NCBI Type	Description
501 sequences.	Alignment		[Dense-seg]	KJ620018.1 (1..401) x AF163582.1 (256..562) x AY026918.1 (300.....

Distance Method: Jukes-Cantor (DNA)
Tree Construct Method: Neighbor Joining
Labels for Leaf Nodes: Taxonomic Name (if available)

Fig. 21 Run Tool: Phylogenetic Tree Builder

that may contain information about the source, such as BioProject, BioSample, etc.

- How the sequence data was constructed—what technologies were used, how deep was the coverage, etc.
- Which molecules are being represented—was the item that was sequenced a portion of a chromosome? An organelle? An RNA transcript? A protein?
- Annotation of the sequence—which nucleotide positions are transcribed to form RNA? Which positions in the transcription are translated to produce proteins? Which positions in the transcription represent the final mature peptide? Which positions represent regulatory elements?

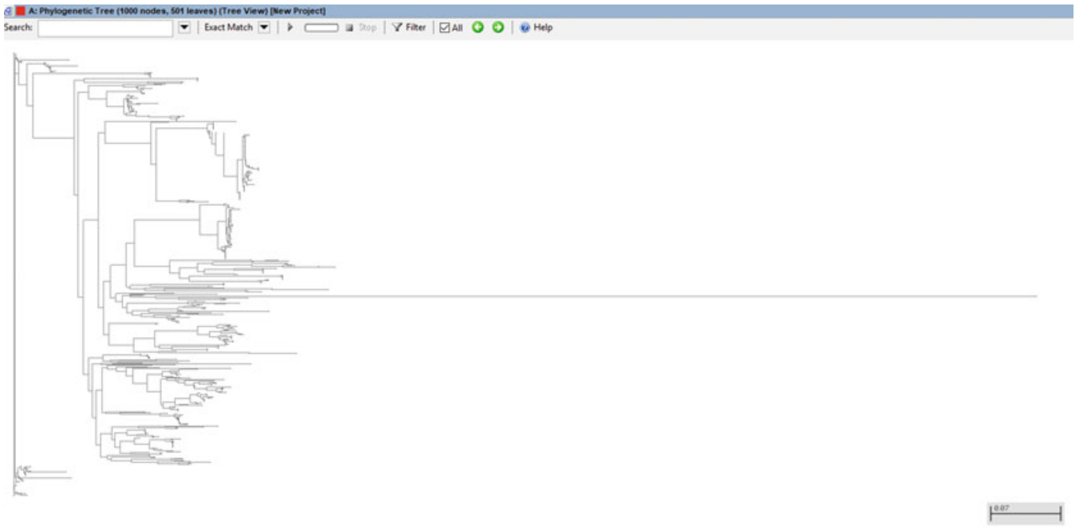

A: Phylogenetic Tree (1000 nodes, 501 leaves) (Tree View) [New Project]
Search:
Exact Match
Stop
Filter
All
Help
0.07

Fig. 22 Genome Workbench Tree View can show results of computational tools or experiments or trees loaded from external data like Newick

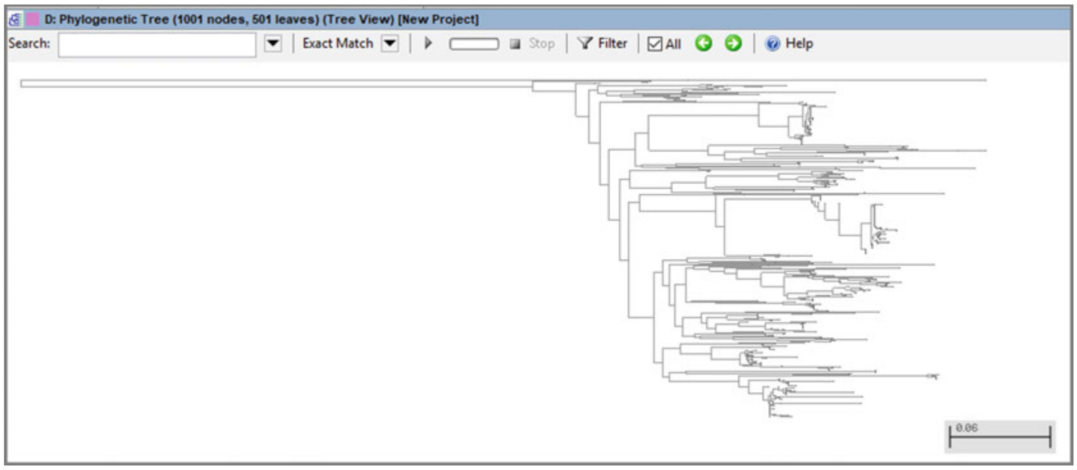

D: Phylogenetic Tree (1001 nodes, 501 leaves) (Tree View) [New Project]
Search:
Exact Match
Stop
Filter
All
Help
0.06

Fig. 23 Tree View shows topology with new tree root (midpoint)

The user may open an existing FASTA or ASN.1 file with the File Open dialog and then start the Genome Submission Wizard, or simply start with the Genome Submission Wizard. The Genome Submission Wizard will prompt the user to open a FASTA or ASN.1 file if no file has been opened. The Genome Submission Wizard will import information from the existing data and guide the user to provide any missing information.

Note that some of the information entered into the Genome Submission Wizard can be exported and imported as a template file. Template files can also be generated using this webpage: https://submit.ncbi.nlm.nih.gov/genbank/template/submission/

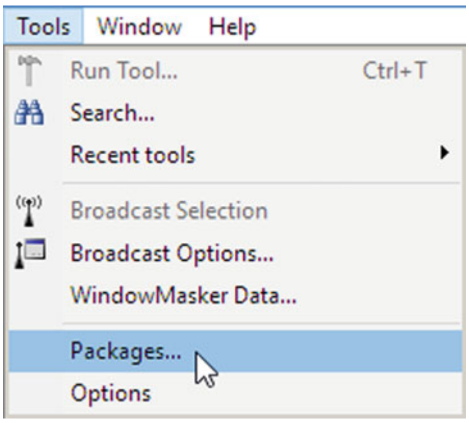
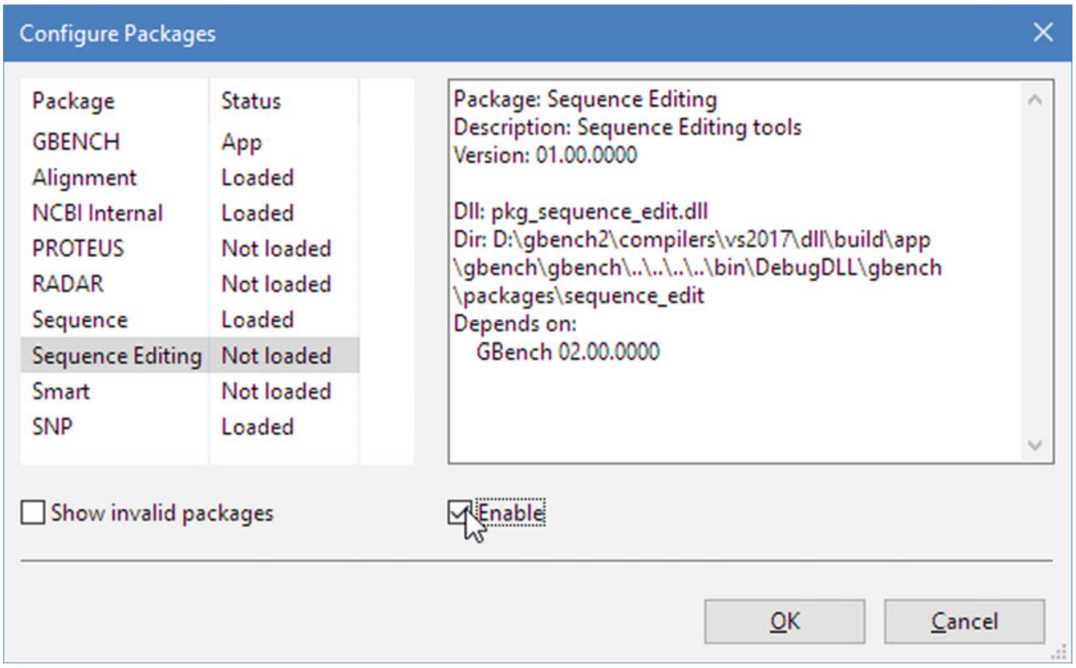


Fig. 24 Selecting Packages from the Tools menu


Configure Packages

Package	Status
GBENCH	App
Alignment	Loaded
NCBI Internal	Loaded
PROTEUS	Not loaded
RADAR	Not loaded
Sequence	Loaded
Sequence Editing	Not loaded
Smart	Not loaded
SNP	Loaded

```
Package: Sequence Editing
Description: Sequence Editing tools
Version: 01.00.0000

Dll: pkg_sequence_edit.dll
Dir: D:\gbench2\compilers\vs2017\dll\build\app
\gbench\gbench\..\..\..\..\bin\DebugDLL\gbench
\packages\sequence_edit
Depends on:
   GBench 02.00.0000
```

☐ Show invalid packages
☑ Enable

Fig. 25 Enabling the Sequence Editing package

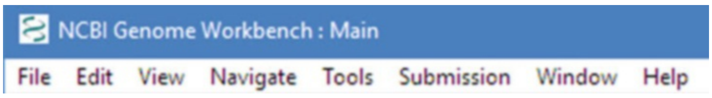


Fig. 26 Submission menu is now available

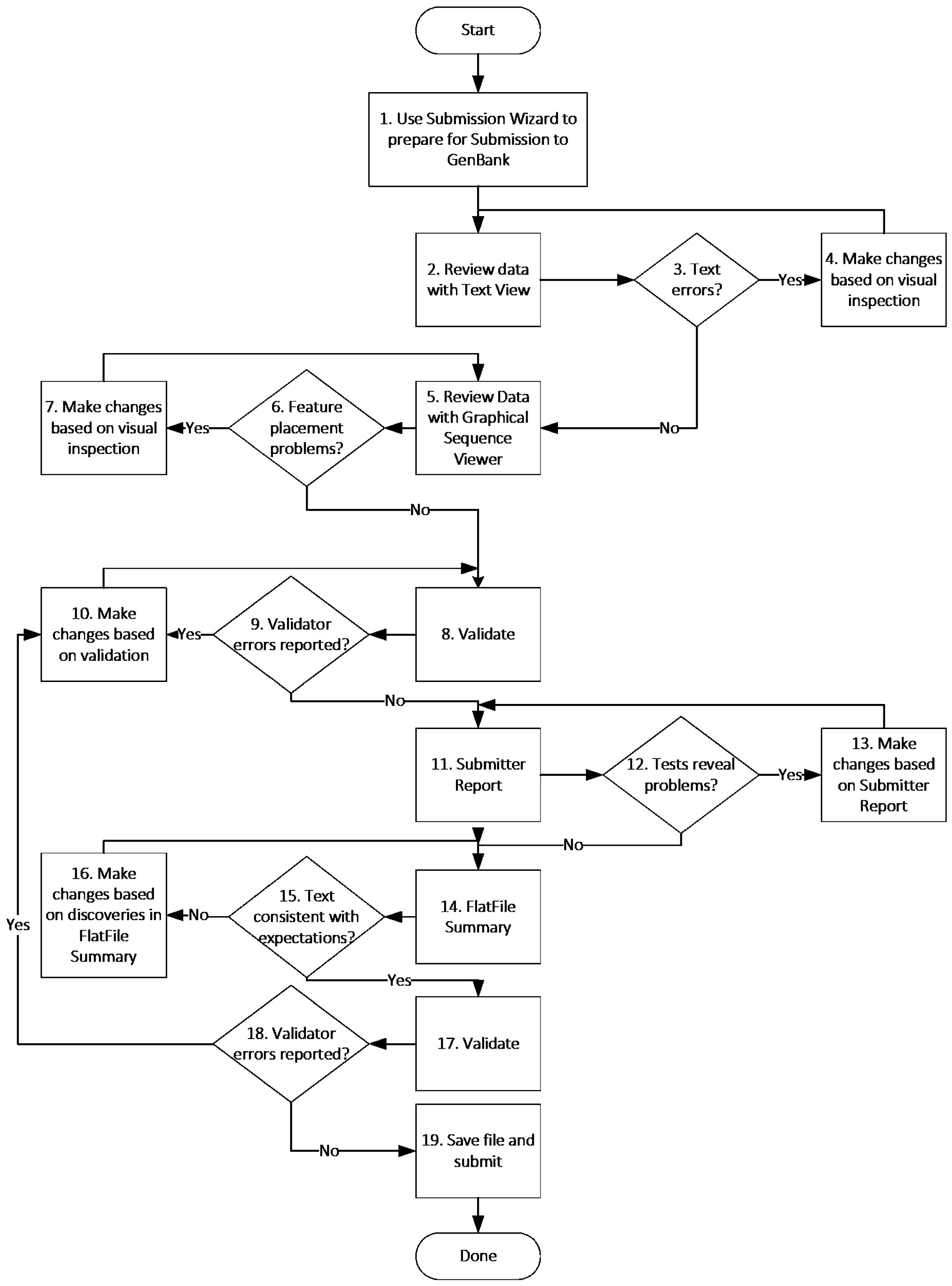
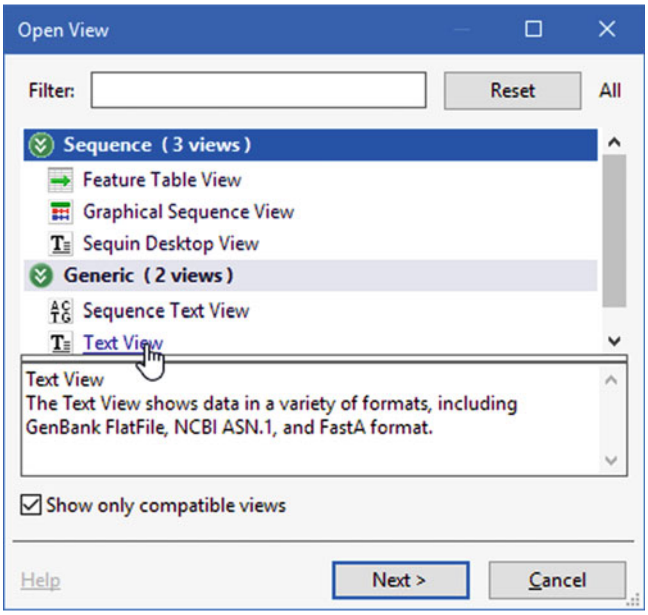
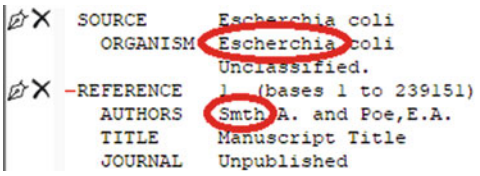

Start
1. Use Submission Wizard to prepare for Submission to GenBank
2. Review data with Text View
3. Text errors?
Yes
4. Make changes based on visual inspection
No
5. Review Data with Graphical Sequence Viewer
6. Feature placement problems?
Yes
7. Make changes based on visual inspection
No
8. Validate
9. Validator errors reported?
Yes
10. Make changes based on validation
No
11. Submitter Report
12. Tests reveal problems?
Yes
13. Make changes based on Submitter Report
No
14. FlatFile Summary
15. Text consistent with expectations?
No
16. Make changes based on discoveries in FlatFile Summary
Yes
17. Validate
18. Validator errors reported?
Yes
No
19. Save file and submit
Done

Fig. 27 Genome Workbench Submission Editing Workflow Process



Text View
The Text View shows data in a variety of formats, including GenBank FlatFile, NCBI ASN.1, and FastA format.

Fig. 28 Creating a Text View



```
SOURCE      Escherchia coli
  ORGANISM  Escherchia coli
            Unclassified.
REFERENCE   1  (bases 1 to 239151)
  AUTHORS   Smth,A. and Poe,E.A.
  TITLE     Manuscript Title
  JOURNAL   Unpublished
```

Fig. 29 Flat File showing misspelled organism name and author name

2. **Review Data with Text View**
   Use the Text View as a first look at the sequence data. The Text View will show how the sequence data will be displayed on the NCBI website. If a Text View is not already open, one will be launched automatically by the Genome Submission Wizard after an ASN.1 or FASTA file has been opened. To launch a new Text View, invoke the Open New View item in the View menu. Choose Text View (Fig. 28).
   More information about the Text View can be found here: https://www.ncbi.nlm.nih.gov/tools/gbench/tutorial25/
3. **Text Errors?**
   Are the author names spelled correctly and in the correct order? Is source information complete and correct? (Fig. 29)
4. **Make Changes Based on Visual Inspection**
   When the Text View is showing the Flat File format and the Sequence Editing Package is enabled, the user can edit the data represented by the Flat File. The pen icon in the left margin

allows the user to launch a dialog to edit the portion of the data that generates that portion of the Flat File. Note that some text cannot be edited directly because the wording is calculated based on the content of the sequence, for example, "BASE COUNT" line shows the number of A, T, G, C, and ambiguity characters present, and cannot be edited directly, although the user can edit the sequence itself and change the content by double-clicking on the lines of sequence after "ORIGIN." The X icon in the left margin allows the user to delete the portion of the data that generates that portion of the Flat File. Note that some portions of the Flat File have default values for when data is not present, so deleting the item will not necessarily remove the section from the Flat File completely.

By default, the Flat File shows all the nucleotide sequences, but the user can also choose to view the Flat File for a specific sequence or for all nucleotide and protein sequences by using the Sequence(s) control next to the configuration icon. Users can also select a sequence to view using the Submission- > Editing Tools- > Select Specific Sequence by Sequence ID menu item. When using Submission menu items that act on a specific sequence, the action will apply to either the single sequence being viewed or the only nucleotide sequence in the record.

5. **Review Data with Graphical Sequence Viewer**

   Launch the Graphical Sequence Viewer to inspect the features that have been annotated on the sequence. In the View menu, choose Open New View and select the Graphical Sequence View (Fig. 30).

   More information about the Graphical Sequence Viewer can be found here:

   https://www.ncbi.nlm.nih.gov/tools/gbench/tutorial23/

6. **Feature Placement Problems?**

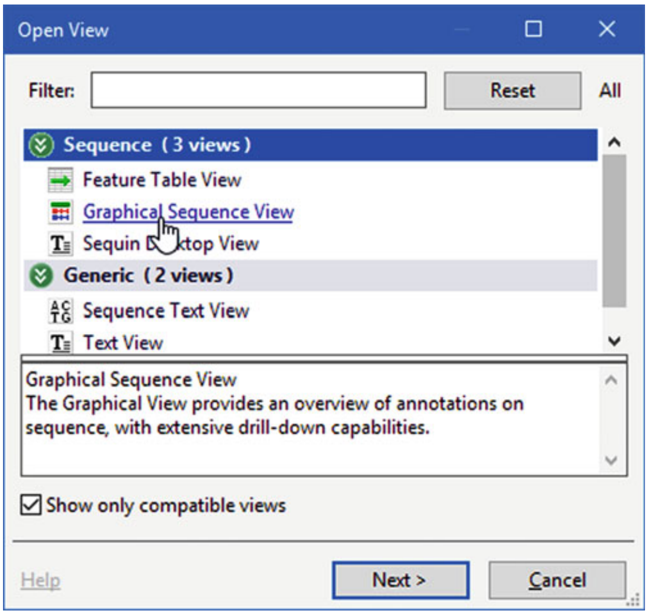
   Are the placements correct? Do features overlap that should not? For example, transfer RNA (tRNA) features and ribosomal RNA (rRNA) features should not overlap (Fig. 31).

7. **Make Changes Based on Visual Inspection**

   Selecting an item in the Graphical Sequence View will also select it in the Text View. From the Text View, click on the pen icon next to the selected feature to launch an editor in order to change the location of the feature, or correct other information about the feature—perhaps this feature was mistakenly created as the wrong type.

8. **Validate**

   To validate the record, invoke the Submission- > Reports- > Validation Report menu item.



Graphical Sequence View
The Graphical View provides an overview of annotations on sequence, with extensive drill-down capabilities.

Fig. 30 Creating a Graphical Sequence View

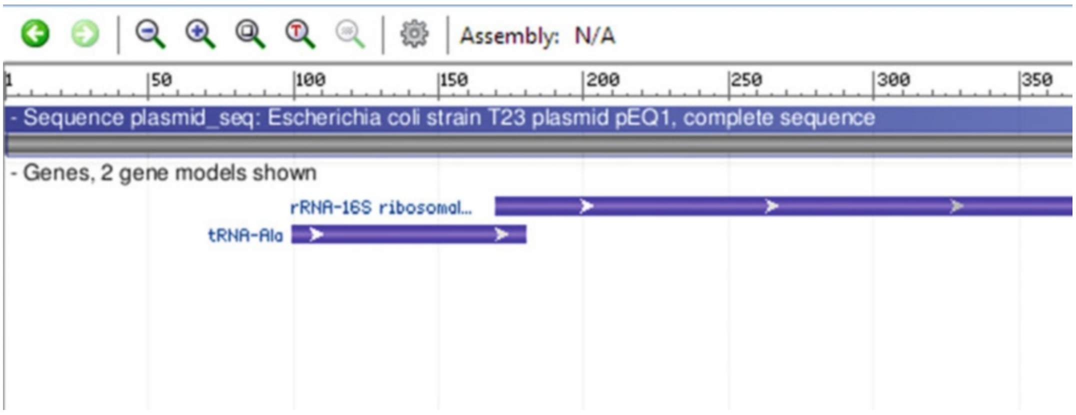
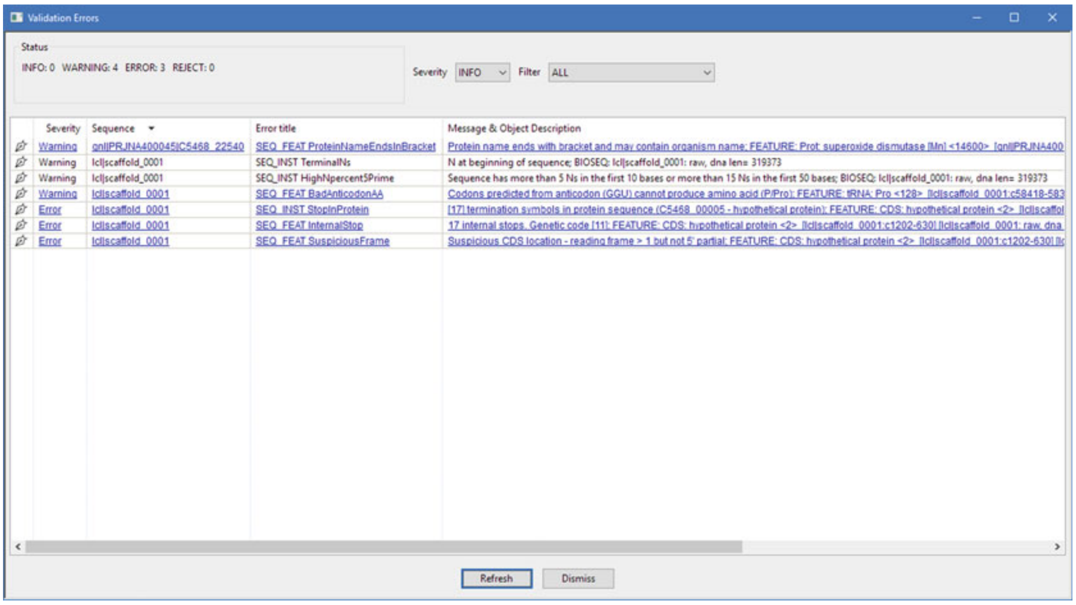

Assembly: N/A
1
50
100
150
200
250
300
350
- Sequence plasmid_seq: Escherichia coli strain T23 plasmid pEQ1, complete sequence
- Genes, 2 gene models shown
rRNA-16S ribosomal...
tRNA-Ala

Fig. 31 Graphical Sequence View showing overlapping tRNA and rRNA features

Genome Workbench provides two tools to help users look for problems with a file that is being prepared for submission to GenBank, the Validator and the Submitter Report. The Validator is focused on individual items that have problems, while the Submitter Report provides information about patterns in the submission. Not all items reported by the Submitter Report are necessarily a sign of a problem. For example, the Submitter Report will list the number of coding regions that are present, which can be compared to the user's expectations.



Status

INFO: 0 WARNING: 4 ERROR: 3 REJECT: 0

Severity INFO Filter ALL

	Severity	Sequence	Error title	Message & Object Description
	Warning	gnl|PRJNA400045|C5468_22540	SEQ_FEAT ProteinNameEndsInBracket	Protein name ends with bracket and may contain organism name: FEATURE: Prot: superoxide dismutase [Mn] <14600> [gnl|PRJNA400
	Warning	lcl|scaffold_0001	SEQ_INST TerminalNs	N at beginning of sequence; BIOSEQ: lcl|scaffold_0001: raw, dna len= 319373
	Warning	lcl|scaffold_0001	SEQ_INST HighNpercent5Prime	Sequence has more than 5 Ns in the first 10 bases or more than 15 Ns in the first 50 bases; BIOSEQ: lcl|scaffold_0001: raw, dna len= 319373
	Warning	lcl|scaffold_0001	SEQ_FEAT BadAnticodonAA	Codons predicted from anticodon (GGU) cannot produce amino acid (P/Pro): FEATURE: tRNA: Pro <128> [lcl|scaffold_0001:c58418-583
	Error	lcl|scaffold_0001	SEQ_INST StopInProtein	[17] termination symbols in protein sequence (C5468_00005 - hypothetical protein): FEATURE: CDS: hypothetical protein <2> [lcl|scaffol
	Error	lcl|scaffold_0001	SEQ_FEAT InternalStop	17 internal stops. Genetic code [11]: FEATURE: CDS: hypothetical protein <2> [lcl|scaffold_0001:c1202-630] [lcl|scaffold_0001: raw, dna
	Error	lcl|scaffold_0001	SEQ_FEAT SuspiciousFrame	Suspicious CDS location - reading frame > 1 but not 5' partial: FEATURE: CDS: hypothetical protein <2> [lcl|scaffold_0001:c1202-630] [lc

Fig. 32 Validator dialog showing errors and warnings

More information about the Validator can be found here: https://www.ncbi.nlm.nih.gov/tools/gbench/manual8/#validator

9. **Validator Errors Reported?**

   Validator error messages will have a severity, associated sequence, title, and specific message (Fig. 32). Validator messages are flagged with a severity level (REJECT, ERROR, WARNING, INFO). Prior to submission, all REJECT and ERROR level validator messages should be resolved. WARNING and INFO level messages report issues that may, in some instances, be valid so it is not necessary to resolve these before submission.

10. **Make Changes Based on Validation**

    The user can click on any of these items to navigate to the object the message is describing or click on the pen icon in the leftmost column to launch an editor for the offending object or a tool for fixing the error. For example, if a sequence does not have any biological source information, the validator will report error code "NoSourceDescriptor." Clicking on the pen icon for this error will launch a dialog to allow the user to add the biological source information for that sequence. If a sequence has ambiguity (N) characters at either end, the validator will report the error code "TerminalNs." Clicking on the pen icon for this error will launch a dialog to trim the ambiguous characters from the ends of all sequences in the file. Note that this action could resolve multiple error messages, as it would affect all sequences with this problem. Also note that when trimming these ambiguity characters, the features that have already been annotated on the sequence will be adjusted so that they still cover the same relative nucleotide positions. The validator reports semantic and syntactic errors and are generally indicative of problems with the data in the submission. For example, a coding region that contains stop codons in the open reading frame and cannot produce a valid protein or a source qualifier value does not conform to INSDC syntax.

The validator will not automatically update after the user has edited the data. The user can hit the "Refresh" to see the remaining problems.

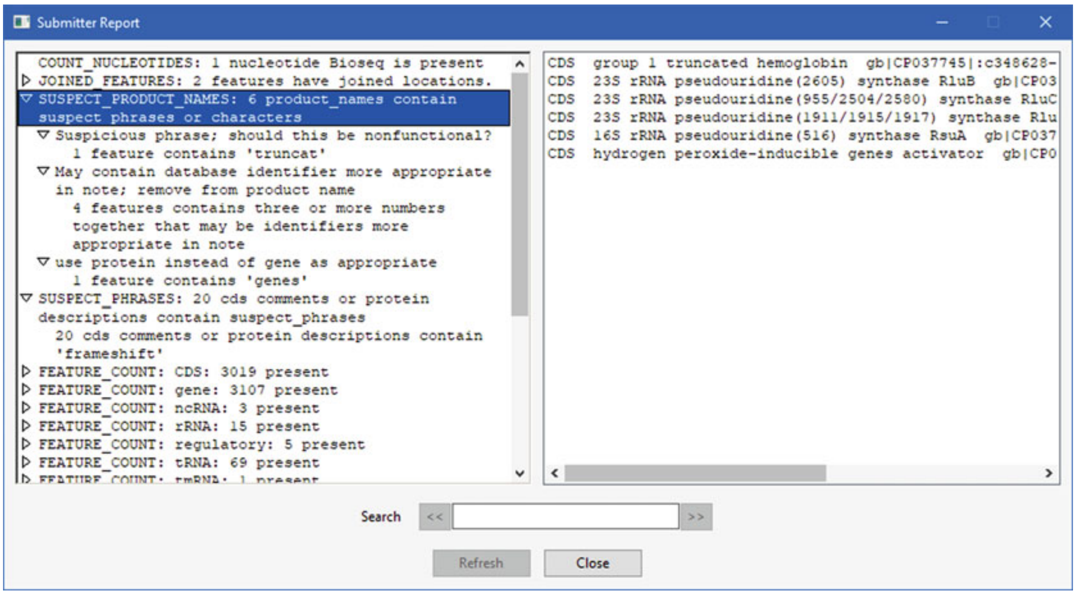
11. **Submitter Report**

To launch the Submitter Report, invoke the Submission- > Reports- > Submitter Report menu item.

The Submitter Report is a tool to help users look for patterns and potential problems with a file. It performs a set of tests and lists the items flagged by each test.

The Submitter Report dialog consists of two panels at the top, a text search box, and some additional buttons at the bottom (Fig. 33). The panel on the left lists the tests for which there are results.

The Search control below the two panels can be used to find text in the left panel.



```
  COUNT_NUCLEOTIDES: 1 nucleotide Bioseq is present
▷ JOINED_FEATURES: 2 features have joined locations.
▽ SUSPECT_PRODUCT_NAMES: 6 product_names contain
  suspect phrases or characters
  ▽ Suspicious phrase; should this be nonfunctional?
      1 feature contains 'truncat'
  ▽ May contain database identifier more appropriate
    in note; remove from product name
      4 features contains three or more numbers
      together that may be identifiers more
      appropriate in note
  ▽ use protein instead of gene as appropriate
      1 feature contains 'genes'
▽ SUSPECT_PHRASES: 20 cds comments or protein
  descriptions contain suspect_phrases
    20 cds comments or protein descriptions contain
    'frameshift'
▷ FEATURE_COUNT: CDS: 3019 present
▷ FEATURE_COUNT: gene: 3107 present
▷ FEATURE_COUNT: ncRNA: 3 present
▷ FEATURE_COUNT: rRNA: 15 present
▷ FEATURE_COUNT: regulatory: 5 present
▷ FEATURE_COUNT: tRNA: 69 present
▷ FEATURE_COUNT: tmRNA: 1 present
```

```
CDS  group 1 truncated hemoglobin  gb|CP037745|:c348628-
CDS  23S rRNA pseudouridine(2605) synthase RluB  gb|CP03
CDS  23S rRNA pseudouridine(955/2504/2580) synthase RluC
CDS  23S rRNA pseudouridine(1911/1915/1917) synthase Rlu
CDS  16S rRNA pseudouridine(516) synthase RsuA  gb|CP037
CDS  hydrogen peroxide-inducible genes activator  gb|CP0
```

Fig. 33 Submitter Report showing test results

More information about the Submitter Report can be found here:

https://www.ncbi.nlm.nih.gov/tools/gbench/manual8/#submitter_report

12. **Tests Reveal Problems?**

For example, the Submitter Report will list product names that may be uninformative, misspelled, contain text that would be more appropriate in a different field, or have other problems using the "SUSPECT_PRODUCT_NAMES" test. Not all items reported by the Submitter Report are necessarily a sign of a problem. For example, the "FEATURE_COUNT" test will list the number of each type of features that are present, which can be compared to the user's expectations. Information about how to interpret individual tests and how to fix problems (when appropriate) can be found here: https://www.ncbi.nlm.nih.gov/genbank/asndisc.examples/

13. **Make Changes Based on Submitter Report**

Clicking on an item in the left panel will cause the results for the test to be displayed in the panel on the right. For tests that refer to coding regions, RNA features, genes, or biological source information, double-clicking on the panel on the left will launch a Bulk Editor to help the user edit the affected items. The user may also click on individual items in the panel on the right to navigate to the item, or double-click on the item to launch an editor for the item.

Note that the Submitter Report does not automatically refresh after editing, so the user must click on Refresh to see the updated results.

14. **FlatFile Summary**

To launch the FlatFile Summary, invoke the Submission- > Reports- > FlatFile Summary menu item.

The FlatFile Summary dialog provides a summary of the nucleotide sequences in the file (Fig. 34). The content is produced by sorting the lines of the FlatFile representations for each nucleotide sequence in the file by section and counting the number of times each line appears identically. For UNIX users, this is similar to applying sort | uniq -c to the FlatFile sections.

The sorted text appears in the top panel of the dialog, grouped into the appropriate sections. These sections can be expanded to show the actual lines of text and the number of times that particular line appears.

More information about the FlatFile Summary can be found here:

https://www.ncbi.nlm.nih.gov/tools/gbench/manual8/#flat_file_summary

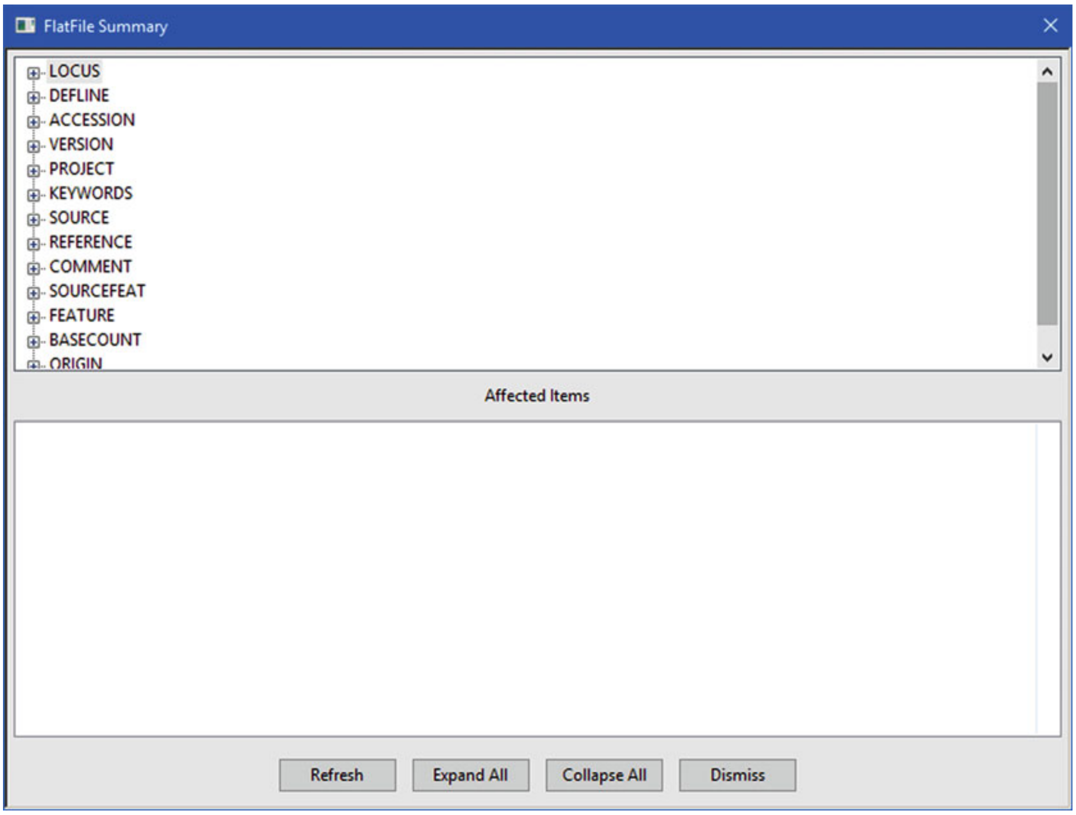


Fig. 34 FlatFile Summary dialog

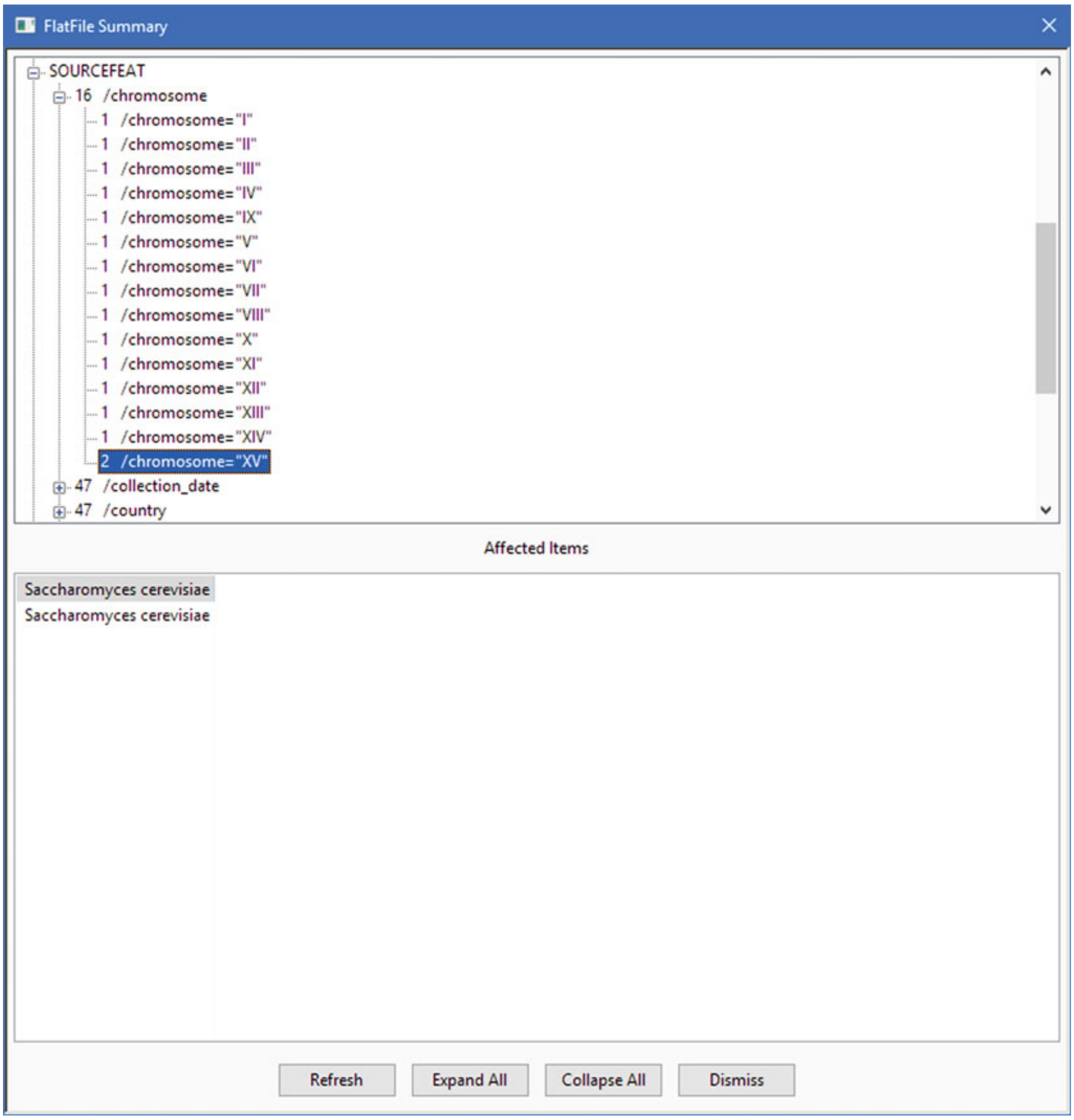
15. **Text Consistent with Expectations?**

    The FlatFile Summary tool is designed to help users quickly look for consistency and find unexpected duplicates. For example, a user might want to confirm that all sequences in a genome submission have the same organism name, but each sequence has a different chromosome value (Fig. 35).

16. **Make Changes Based on Discoveries in FlatFile Summary**

    Clicking on a line of text will cause a list of the items that contain this text to appear in the bottom panel. Clicking on the item in the list will navigate to that item in the Text View, and double-clicking on the item will launch an editing dialog for the item.

    The tool can also be a convenient mechanism for navigating a large record. For example, if the user wants to examine all of the features for which an Enzyme Commission Number (EC number) has been assigned, the user could expand the FEATURE section and look at the /EC_number section to see the list of features, and click on them one at a time (Fig. 36).



FlatFile Summary

SOURCEFEAT
16 /chromosome
1 /chromosome="I"
1 /chromosome="II"
1 /chromosome="III"
1 /chromosome="IV"
1 /chromosome="IX"
1 /chromosome="V"
1 /chromosome="VI"
1 /chromosome="VII"
1 /chromosome="VIII"
1 /chromosome="X"
1 /chromosome="XI"
1 /chromosome="XII"
1 /chromosome="XIII"
1 /chromosome="XIV"
2 /chromosome="XV"
47 /collection_date
47 /country

Affected Items

Saccharomyces cerevisiae
Saccharomyces cerevisiae

Fig. 35 FlatFile Summary showing chromosome values

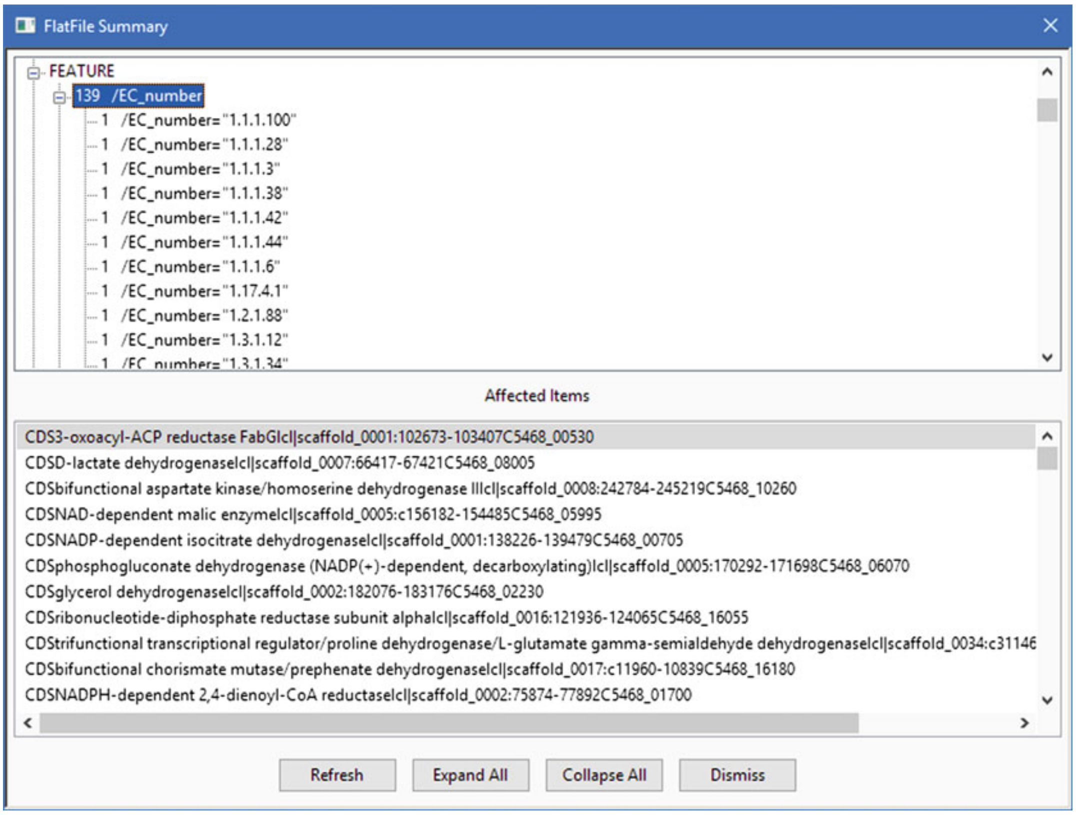
Note that the FlatFile Summary is not automatically updated after changing the data, so the user should be sure to use the Refresh button to incorporate the most recent changes.

17. **Validate**

    This is a final check for errors, the same as **step 8**.

18. **Validator Errors Reported?**

    If validator errors were reported, return to **step 10** to correct them.



```
FEATURE
  139  /EC_number
    1  /EC_number="1.1.1.100"
    1  /EC_number="1.1.1.28"
    1  /EC_number="1.1.1.3"
    1  /EC_number="1.1.1.38"
    1  /EC_number="1.1.1.42"
    1  /EC_number="1.1.1.44"
    1  /EC_number="1.1.1.6"
    1  /EC_number="1.17.4.1"
    1  /EC_number="1.2.1.88"
    1  /EC_number="1.3.1.12"
    1  /EC_number="1.3.1.34"
```

Affected Items

```
CDS3-oxoacyl-ACP reductase FabGlcl|scaffold_0001:102673-103407C5468_00530
CDSD-lactate dehydrogenaselcl|scaffold_0007:66417-67421C5468_08005
CDSbifunctional aspartate kinase/homoserine dehydrogenase IIIcl|scaffold_0008:242784-245219C5468_10260
CDSNAD-dependent malic enzymelcl|scaffold_0005:c156182-154485C5468_05995
CDSNADP-dependent isocitrate dehydrogenaselcl|scaffold_0001:138226-139479C5468_00705
CDSphosphogluconate dehydrogenase (NADP(+)-dependent, decarboxylating)lcl|scaffold_0005:170292-171698C5468_06070
CDSglycerol dehydrogenaselcl|scaffold_0002:182076-183176C5468_02230
CDSribonucleotide-diphosphate reductase subunit alphalcl|scaffold_0016:121936-124065C5468_16055
CDStrifunctional transcriptional regulator/proline dehydrogenase/L-glutamate gamma-semialdehyde dehydrogenaselcl|scaffold_0034:c31146
CDSbifunctional chorismate mutase/prephenate dehydrogenaselcl|scaffold_0017:c11960-10839C5468_16180
CDSNADPH-dependent 2,4-dienoyl-CoA reductaselcl|scaffold_0002:75874-77892C5468_01700
```

Fig. 36 FlatFile Summary showing features with EC numbers

19. **Save File and Submit**

    Use the Submission- > Save Submission File menu item to save your file and submit the file via the NCBI Submission Portal (https://submit.ncbi.nlm.nih.gov/subs/genome/). Choose "Single" or "Batch/multiple" genomes. Answer the questions and upload the necessary files. Review the summary page and click the "Submit" button. The submission will be given a "SUB" temporary identifier which can be used in correspondence before an accession number is assigned to the genome submission.

Acknowledgments

This research was supported by the Intramural Research Program of the National Library of Medicine, National Institutes of Health.

References

1. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25(17):3389–3402. https://doi.org/10.1093/nar/25.17.3389
2. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL (2009) BLAST+: architecture and applications. BMC Bioinformatics 10:421. https://doi.org/10.1186/1471-2105-10-421
3. Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32(5):1792–1797. https://doi.org/10.1093/nar/gkh340
4. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Soding J, Thompson JD, Higgins DG (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
5. Sayers EW, Cavanaugh M, Clark K, Ostell J, Pruitt KD, Karsch-Mizrachi I (2019) GenBank. Nucleic Acids Res 47(D1):D94–D99. https://doi.org/10.1093/nar/gky989
6. O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, McVeigh R, Rajput B, Robbertse B, Smith-White B, Ako-Adjei D, Astashyn A, Badretdin A, Bao Y, Blinkova O, Brover V, Chetvernin V, Choi J, Cox E, Ermolaeva O, Farrell CM, Goldfarb T, Gupta T, Haft D, Hatcher E, Hlavina W, Joardar VS, Kodali VK, Li W, Maglott D, Masterson P, McGarvey KM, Murphy MR, O'Neill K, Pujar S, Rangwala SH, Rausch D, Riddick LD, Schoch C, Shkeda A, Storz SS, Sun H, Thibaud-Nissen F, Tolstoy I, Tully RE, Vatsan AR, Wallin C, Webb D, Wu W, Landrum MJ, Kimchi A, Tatusova T, DiCuccio M, Kitts P, Murphy TD, Pruitt KD (2016) Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Res 44(D1):D733–D745. https://doi.org/10.1093/nar/gkv1189
7. Desper R, Gascuel O (2002) Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. J Comput Biol 9(5):687–705. https://doi.org/10.1089/10665270276 1034136
8. Desper R, Gascuel O (2004) Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. Mol Biol Evol 21(3):587–598. https://doi.org/10.1093/molbev/msh049
9. Ciufo S, Kannan S, Sharma S, Badretdin A, Clark K, Turner S, Brover S, Schoch CL, Kimchi A, DiCuccio M (2018) Using average nucleotide identity to improve taxonomic assignments in prokaryotic genomes at the NCBI. Int J Syst Evol Microbiol 68(7):2386–2392. https://doi.org/10.1099/ijsem.0.002809

Part IV

Open Problems

Chapter 17

Revisiting Evaluation of Multiple Sequence Alignment Methods

Tandy Warnow

Abstract

Multiple sequence alignment is a core first step in many bioinformatics analyses, and errors in these alignments can have negative consequences for scientific studies. In this article, we review some of the recent literature evaluating multiple sequence alignment methods and identify specific challenges that arise when performing these evaluations. In particular, we discuss the different trends observed in simulation studies and when using biological benchmarks. Overall, we find that multiple sequence alignment, far from being a "solved problem," would benefit from new attention.

Key words Multiple sequence alignment, Phylogeny estimation, Model misspecification, Structural alignment, Statistical alignment

1 Introduction

Multiple sequence alignment (MSA) is a preliminary step in much biological research, including phylogeny estimation, protein structure and function prediction, sequence classification into gene families, and even genome assembly. Yet MSA estimation is known to be difficult under many conditions, and errors in estimated MSAs can lead to errors in downstream analyses, such as phylogeny estimation [1–5]. Because of the impact of MSA estimation, the development of new MSA methods and their evaluation is of interest to a large number of practitioners.

The focus of this chapter is on multiple sequence alignment accuracy, and specifically assumes that the "correct" alignment is one where all columns represent true positional homology (i.e., descent from a common ancestor). Since alignments are used in many applications, we also address the degree to which alignment accuracy correlates with accuracy for the downstream application (e.g., phylogeny estimation, protein structure, etc.). One of the specific issues we confront is the difficulty in obtaining meaningful benchmarks with which to evaluate estimated alignments: reference alignments based on simulated datasets are accurate but potentially insufficiently realistic, and biological datasets are by definition realistic but the reference alignments may not be completely accurate. This is an issue we will address several times during the chapter.

Another issue we will address is the difficulty in quantifying accuracy (or error, the complementary view) in an estimated alignment so that it is predictive of accuracy for downstream analyses. For example, many standard criteria do not correlate with each other (e.g., some alignments can be good for one criterion and poor for another), and none of them are that well correlated with phylogenetic accuracy, one of the major downstream analyses. Our overall findings are that the evaluation of MSA methods is substantially complex, and that current approaches to evaluating methods are too simplistic to adequately characterize methods. Our study also suggests directions for future research in multiple sequence alignment as well as other statistical estimation problems in biology.

The rest of the chapter is organized as follows. We begin with terminology in Subheading 2. We discuss results from the literature on multiple sequence alignment evaluation in Subheading 3. We conclude with a discussion of the trends revealed in these studies and directions for future work in Subheading 4.

2 Overview

Throughout this section, we will assume that the input contains the estimated alignment $\hat{o}$ and the true (or reference) alignment A, and we are comparing the two alignments to each other.

2.1 Terminology

Every multiple sequence alignment can be seen as a matrix where the rows correspond to the sequences and the columns represent homology, which is the same as saying that the letters (i.e., nucleotides or amino acids) in the column are all derived from a common ancestor [6]. This definition applies to both nucleotide and protein alignments, even though some researchers have used "homology" to refer to structural or functional similarity [7]. As a result, every alignment can be described by its set of "homology pairs," which are the pairs of letters that appear in any column. This representation requires that we distinguish between nucleotides or amino acids (residues) based on their position. There are many criteria that have been proposed for evaluating alignment accuracy [6, 8, 9], but here we focus on a subset of these possible criteria that are in common use.

2.2 Standard Alignment Criteria

We can evaluate the accuracy (or error) of an estimated alignment $\hat{o}$ by comparing its set of homology pairs to the set of homology pairs for the true alignment A. Note that every homology pair in an estimated alignment is either a true positive (if it appears also in the true alignment) or a false positive (if it does not appear in the true alignment). Similarly, the homology pairs in the true alignment that do not appear in the estimated alignment are called *false negatives*. Furthermore, the *false negative rate* is the fraction of the homology pairs in the true alignment that are false negatives (i.e., missing from the estimated alignment), and the *false positive rate* is the fraction of the homology pairs in the estimated alignment that are false positives. Note that these are error rates, and so are values in the range [0, 1]. We refer to these as SPFN (sum-of-pairs false negatives) and SPFP (sum-of-pairs false positives).

In the protein alignment world, the standard criteria for evaluating alignments are phrased in terms of accuracy rather than error, and are as follows:

- SP-score = 1-SPFN, which is a measure of recall; the best score is 1.0.
- Modeler score = 1-SPFP, which is a measure of precision; the best score is 1.0.
- Total Column (TC) score: the number of columns in the estimated alignment that appear in the same exact form in the true alignment; the best score is the number of columns in the true alignment.
- Expansion ratio: the ratio between the length of the estimated alignment and the length of the true or reference alignment, and so the best score is 1.0.

2.3 Evolutionary Criteria

When phylogeny estimation is the focus of the study, then criteria that evaluate phylogeny estimation accuracy are commonly considered. When sequences are simulated down a model tree, then the accuracy of a tree computed on the estimated alignment can be used to evaluate the alignment accuracy. Similarly, other evolutionary parameters, such as the insertion and deletion rates, the relative frequency of insertions to deletions, and the gap length distribution (of the indel process), or reconstructed ancestral sequences, can also be used to evaluate alignments [10, 11].

2.4 MSA Methods

There are many different MSA methods, each typically based on a combination of techniques, and focused (in some cases) on different objectives. Most methods can be used for both protein and nucleotide alignment, but some can only be used for protein sequence alignment (e.g., SATCHMO [12, 13] and PROMALS [14]). Many methods use progressive alignment techniques and build alignments using a guide tree (which they may also compute from the input).

One class of methods assumes an explicit parametric model of sequence evolution that includes insertions and deletions, as well as substitutions. Examples of such methods include BAli-Phy [15, 16] and StatAlign [17], which are methods that try to co-estimate the alignment and tree under the assumed evolutionary model. Prank [10, 18], PAGAN [19], and Historian [11] are similar in spirit to these methods in that they are based on statistical models, but do not make a full effort to search tree space; instead, they use a given guide tree and compute the alignment on the guide tree, under the assumed evolutionary model.

A complementary approach to MSA estimation uses divide-and-conquer: a large dataset of unaligned sequences is divided into subsets, alignments are computed on subsets, and then merged into an alignment on the full dataset. If desired, a tree can be computed on the dataset, and then used to produce another decomposition of the sequences into subsets and the divide-and-conquer re-alignment strategy can be repeated. These approaches, which include SATé-I [4], SATé-2 [20], and PASTA [21], are best seen as "meta-methods" since they can be used with any selected MSA method. These methods also improve the accuracy of many MSA methods on large challenging datasets (as evaluated on simulated datasets), and enable some methods to run on large datasets where they are unable to otherwise [22].

3 Results from the Literature

In this section, we discuss some of the trends that have been observed in the literature, focusing on the challenges in evaluating alignment methods with respect to accuracy and impact on downstream analyses. As we will show, the relative performance of methods depends very much on the specific details, including the choice of data, the criterion used to compare methods, and even the specific command and version number for the methods.

3.1 Method Performance Depends on Specific Commands and Version Numbers

One of the basic observations about evaluating alignment methods is that the accuracy and running time depends on the specific version and command used. This is particularly true of methods that are under rapid development, so that different versions have different performance. For this reason, it is important to avoid drawing broad conclusions about methods based on older studies, as the methods may have improved since the study was performed.

However, even for a specific version, the accuracy of a method can depend on the specific command. For example, MAFFT [23–26], which is generally considered one of the leading methods, can be run in many ways, and the accuracy depends on the specific commands and dataset properties [25]. In addition, performance studies evaluating MAFFT are often performed using it in *-auto* mode, which allows the program to select the command, with less computationally expensive variants used for large datasets. However, on large datasets, the *-auto* setting for MAFFT is less accurate than some of the more computationally intensive commands, such as MAFFT -L-INS-i or MAFFT -G-INS-i [27].

Similarly, many methods require guide trees to compute their alignments, and will use their own technique if no guide tree is provided. Yet, studies have shown that alignments (and especially trees based on these alignments) can be improved if a carefully selected guide tree is used, instead of the method's default guide tree [28, 29]. Related to this, some studies (e.g., [11, 18]) evaluating alignment methods have been performed using the true tree (either the model tree used to simulate sequences or an established species tree) as the guide tree, which has the potential consequence of improving accuracy in the resultant alignments.

3.2 Impact of Dataset Properties

The relative accuracy between MSA methods may depend on empirical properties of the data, and in particular the average pairwise PID (percent identical across the sites), which is a way of measuring dataset homogeneity, within the dataset can impact the relative performance of methods. For example, Sievers et al. [27] compared a collection of MSA methods on the Prefab [30] benchmark collection, and observed that MSAprobs [31] had the best TC scores of all tested methods when PID was very low (less than 20%) but the worst when PID was very high (at least 70%) (Table II in [27]). They also saw that Clustal-Omega [27] had the best TC scores of all methods when PID was very high (at least 70%) but was only in the top half when PID was very low (at most 20%).

More recent studies on biological benchmarks have also found similar inversions in relative performance as a function of PID. For example, Nute et al. [32] evaluated protein alignment methods on four structural benchmark collections, and found that T-Coffee [33] dominated nearly all other methods for Modeler score and SP-score whenever PID $<$ 50%, but had the poorest scores otherwise. Nute et al. also saw that Clustal-Omega tied for the best Modeler score and SP-score when PID was at least 50% and then tied for worst scores when PID was at most 15%.

The impact of substitution rate on alignment methods has also been evaluated using simulation studies, and confirms these general observations. For example, [32] evaluated alignment methods on simulated protein sequences and found that Prank [10] was the least accurate of the tested methods when mutation rates were high but performed well-coming close to the top performing methods–when mutation rates were low. Similarly, Liu et al. [4] compared alignment methods on large (1000-taxon) simulated nucleotide datasets and found that the relative alignment accuracy of MAFFT and Muscle [30, 34] (both in default mode) changed with the rate of evolution, with Muscle less accurate than MAFFT for low rates of evolution and then more accurate than MAFFT for high rates of evolution.

Compared to the impact of PID (and substitution rate in general), less is known about the impact of other empirical properties on the relative performance of MSA methods. However, Nguyen et al. [35] found that most MSA methods had very poor accuracy on datasets with fragmentary sequences, and that the relative accuracy of PASTA and UPP [35], a method for alignment estimation that uses ensembles of profile Hidden Markov Models, depended on the degree of fragmentation in the dataset (with PASTA generally more accurate than UPP when there was no fragmentation, and UPP consistently more accurate than PASTA when there was a lot of fragmentation).

The number of sequences also impacts the relative performance of methods, in part because some methods do not run on large datasets due to computational or memory requirements, but other factors may also be at play (e.g., it is possible that the guide tree used in progressive alignment methods has a larger impact when the number of sequences is very large, or when the dataset exhibits substantial heterogeneity). It also seems likely that other empirical properties, such as the relative rates of insertions to deletions, gap lengths, and degree of violation of the molecular clock, may also impact the relative performance of methods.

Unfortunately, most studies have inadvertently failed to explore performance under a sufficiently wide range of model conditions for robust inferences to be made about the relative performance of methods. Most studies, for example, simulate sequences where the probabilities of insertion and deletion events are identical (e.g., [4]) or under the strict molecular clock (e.g., [36]); these are conditions that are likely to improve accuracy for most (perhaps all) multiple sequence alignment methods, but may benefit some methods more than others.

3.3 Alignment Criteria Rankings Differ

One of the not infrequent observations in studies evaluating alignment methods is that different criteria rank methods differently. For example, Nute et al. [32] compared different protein alignment methods on various protein structural alignment benchmarks. For the BAliBASE [37] benchmark, BAli-Phy [15, 38, 39] had by far the best Modeler score of all the methods that were examined, but it was in the bottom third for SP-score [32]. The difference between Modeler and SP-score is easily understood, given that one measures precision and the other measures recall, and many statistical tests tend to focus on one but not the other criterion. However, this distinction between criteria indicates there is likely no single criterion that can be used to rank methods.

3.4 Alignment vs. Tree Accuracy

Nearly all phylogenies are based on estimated multiple sequence alignments, so that multiple sequence alignment error has the potential to lead to reduced accuracy in the estimated phylogenies. Many early studies [3, 40–45] examined the impact on both biological and simulated datasets, and observed that alignment error sometimes but not always leads to reduced accuracy, and similarly that the choice of alignment method sometimes but not always has an impact on the final tree.

To reconcile the differences in impact, Wang et al. [5] performed a study evaluating the conditions where alignment accuracy has an impact on phylogenetic accuracy, focusing on protein multiple sequence alignment, mainly using simulations using model trees based on BAliBASE datasets and also birth-death model trees (up to 120 sequences). They varied alignment methods and tree estimation methods, and observed the following trends. First, trees based on maximum likelihood tended to be more accurate than trees based on maximum parsimony (both weighted and unweighted) and neighbor joining (under two ways of calculating distances). Second, the correlation between alignment accuracy and tree accuracy depended on the method for computing trees, with a higher correlation for maximum likelihood (ML) than for maximum parsimony (MP) or neighbor joining (NJ) [46], and also on the model condition (*see* [47] for background on phylogeny estimation). Most significantly, they found that while alignment accuracy and tree accuracy was positively correlated for all model conditions they examined, the correlation was weak when rates of evolution (substitutions and indels) were low, but increased with the rates of evolution. Since alignment error also increased with rates of evolution, they observed that only when alignments were difficult was alignment and tree accuracy strongly correlated. To quote from [5]:

> It seems that only when the estimated alignments are sufficiently poor (perhaps with alignment SP-error rates above 20 or 30 percent) will differences in alignment error reliably produce an appreciable impact on the resultant phylogeny... In summary, there is a generally positive correlation between alignment and tree error when model conditions produce data sets with relatively high average alignment errors. The positive correlation between alignment error and ML tree accuracy is much weaker when alignment errors are low and evaporates for MP and NJ analyses of data sets with low average alignment error. This suggests that except for model conditions that produce data sets that are quite difficult to align, there will only be small consequences to choosing between a very good alignment and a somewhat poorer alignment. Furthermore, data sets with higher evolutionary rates (larger diameters) and more indels tend to show bigger differences in the accuracy of phylogenies estimated on different alignments... For now, we hypothesize that when alignments are relatively easy, there is enough phylogenetic signal in any "reasonable" alignment (even one with perhaps 20 percent of the homologous pairs missing) to reproduce much of the tree one would get if one had the true alignment... These observations may help resolve the seeming contradictory findings of earlier studies, in which alignments have sometimes been shown to have a big impact on phylogenetic estimation, but not always.

These observations have been found in many subsequent studies, including studies evaluating nucleotide alignment on large datasets. For example, Liu et al. [4] examined nucleotide alignment methods on datasets with up to 1000 sequences, and observed that when the rates of evolution were low enough, then most alignment methods had relatively low error (i.e., at most 20%), and trees computed on these estimated alignments were as accurate as trees computed on the true alignment.

Based on these studies, one might assume that if improving alignment accuracy would either be beneficial or at worst neutral for phylogeny estimation. However, other studies have shown surprising inversions, where one method outperformed another with respect to standard alignment accuracy criteria but was worse with respect to phylogenetic accuracy. For example, Liu et al. [4] observed that Prank [10] had higher alignment error than Opal [48] on ribosomal RNA structural alignments from [49] (i.e., Prank had SPFN error of 40.5% compared to 29.3% for Opal) but much lower tree error (14.5% compared to 18.9%). Similarly, a comparison between Opal and SATé-I on the simulated data in [20] showed OPAL had much lower alignment SPFN error than SATé-I, but ML trees on SATé-I alignments had much lower tree error than Opal.

A study by Nelesen et al. [28], which examined the impact of the guide tree in alignment methods, provides additional evidence that alignment accuracy and tree accuracy have a complex relationship (*see* Fig. 1). The natural assumption is that alignments using the true tree as the guide tree should be more accurate than alignments using other guide trees, and furthermore if the guide tree is more accurate (i.e., topologically more similar to the true tree), then alignment accuracy should improve and trees computed on these alignments should also improve. Yet, the study by Nelesen et al. found that this assumption did not hold for many alignment methods. For example, they saw that alignments computed using both ClustalW [51] and Muscle [30, 34] had higher SPFN error when given the true tree as a guide tree than when given a tree computed using UPGMA (a simple distance-based method for computing trees), but trees on these poorer alignments were more accurate. Nelesen et al. also observed the disturbing trend maximum likelihood trees computed on the true alignment were *less accurate* than trees computed on POY [50], when POY was given the true tree as the guide tree. What made this particularly disturbing is that the alignment error produced by POY on the true tree was fairly high (above 20% for the 100-taxon datasets), yet ProbCons [52] on all the tested guide trees had lower alignment error but produced less accurate trees. Thus, POY alignments may not be very accurate but ML trees on POY alignments can be highly accurate.

In general, these studies suggest the possibility that algorithm designs that improve accuracy with respect to standard alignment criteria do not always result in improved trees, and vice versa. It may also be that some methods (perhaps POY, SATé, and PASTA) may exhibit *guide tree imprinting*, so that they produce alignments that tend to reinforce the guide trees they are given.

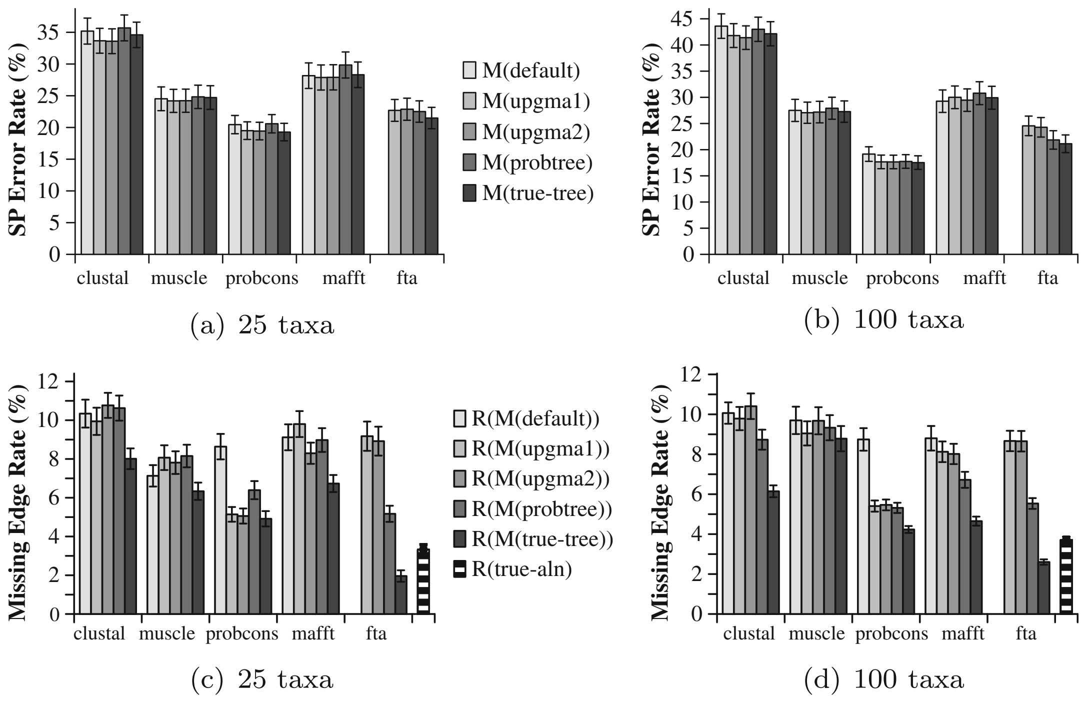


Fig. 1 [Figures 3 and 4 from [28], Copyright 2008, published by World Scientific, reprinted with permission from the publisher] These results are based on a DNA sequence simulation study, using methods for alignments that take guide trees, and then compute RAxML maximum likelihood trees. The top subfigure shows alignment error of different methods using different guide trees, with (**a**, **c**) 25 sequences or (**b**, **d**) 100 sequences, and the bottom subfigure shows the tree error of RAxML trees computed on these alignments. "Default" refers to the guide tree used by the specified alignment method, and "probtree" refers to a maximum likelihood tree computed on the ProbCons alignment. Note that in many cases, changing the guide tree does not impact the alignment error, and that in some cases using the true tree as the guide tree produces higher error than using some other guide tree (e.g., Clustal on the UPGMA guide trees has lower error than Clustal using the true tree). Note also that alignments with essentially the same SP-error can produce trees with very different levels of error (e.g., all ProbCons alignments have nearly the same SP-error, but differ substantially in terms of tree error). Finally, note that FTA (which is POY [50] used to compute an alignment on the specified guide tree) produces a more accurate tree when given the true tree as the guide tree than RAxML on the true alignment; put differently, RAxML on an FTA alignment can be more accurate than RAxML on the true alignment!

3.5 Simulated vs. Biological Datasets

One of the challenges in evaluating alignment methods is that relative performance is not always consistent between simulation studies and biological datasets. A striking example of this phenomenon is reported in Nute et al. [32], where BAli-Phy had the best Modeler and SP-scores of all methods on the simulated datasets for all the tested model conditions, with varying rates of indels and substitutions (Fig. 2), but was generally only in the middle (or in the bottom third, in some cases) of the methods on the protein structural benchmark datasets (Fig. 3).

There are at least three possible explanations for this difference on biological and simulated datasets. One explanation is that the biological benchmarks are incorrect, so that relative performance

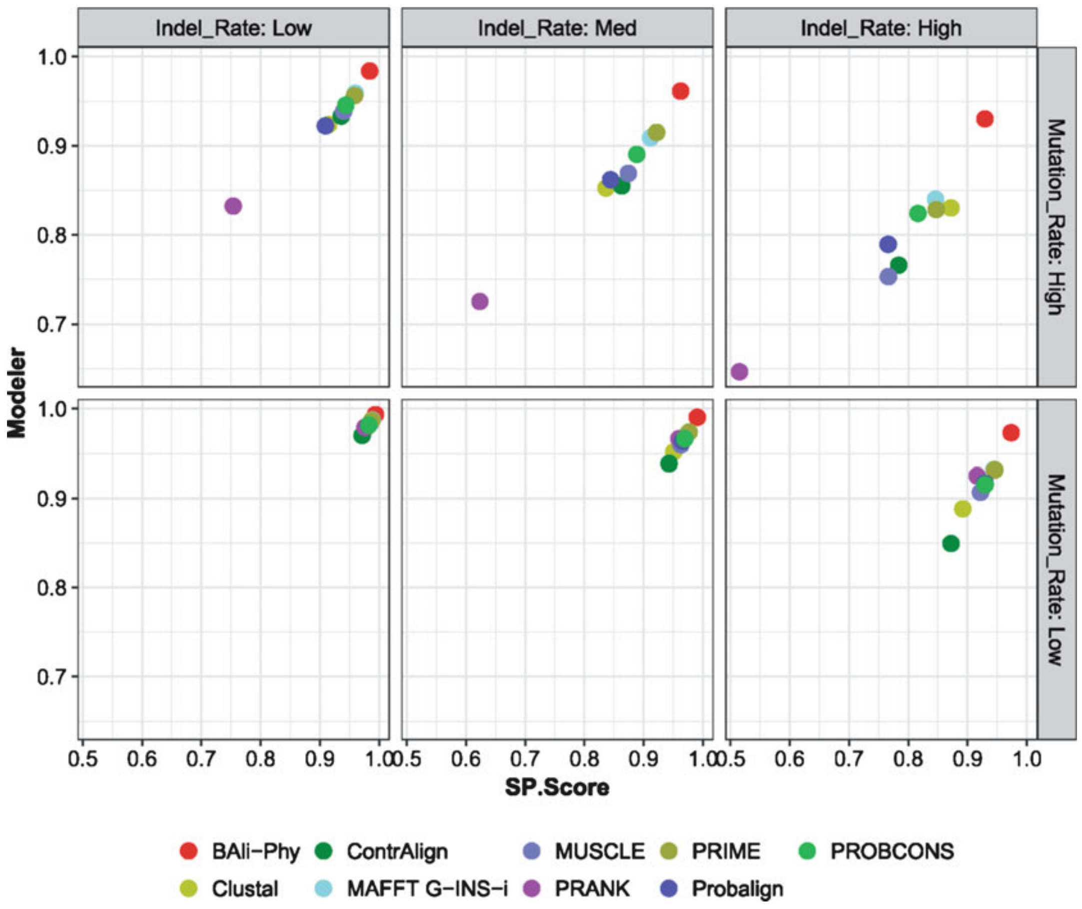


Fig. 2 [Figure 6 from [32], reprinted under the Creative Commons Attribution License (http:/creativecommons.org/licenses/by-nc/4.0/)] This figure shows SP-score and Modeler score for BAli-Phy (using the posterior decoding to produce a single alignment) and several other multiple sequence alignment methods on simulated datasets with 27 sequences under six model conditions, varying in terms of indel rates and substitution rates. Note that BAli-Phy, shown in red, has the best Modeler score (precision) and SP-score (recall) on all of the model conditions

on these datasets is not reflective of actual accuracy. Another explanation is that the biological benchmarks are correct as structural alignments, but having two residues in the same column does not necessarily reflect common evolutionary history (i.e., the structural alignment is not a reflection of shared descent, and so is not a statement of true "homology"). Finally, a third explanation is that the biological benchmarks are correct, even as evolutionary alignments, but that biological evolution is not well described by the model assumed in BAli-Phy. This last explanation would have the possible consequence that accuracy on data simulated under the BAli-Phy model (or similar models) would not reflect likely accuracy on biological data. Most likely all three explanations are valid, but the relative contributions are unknown.

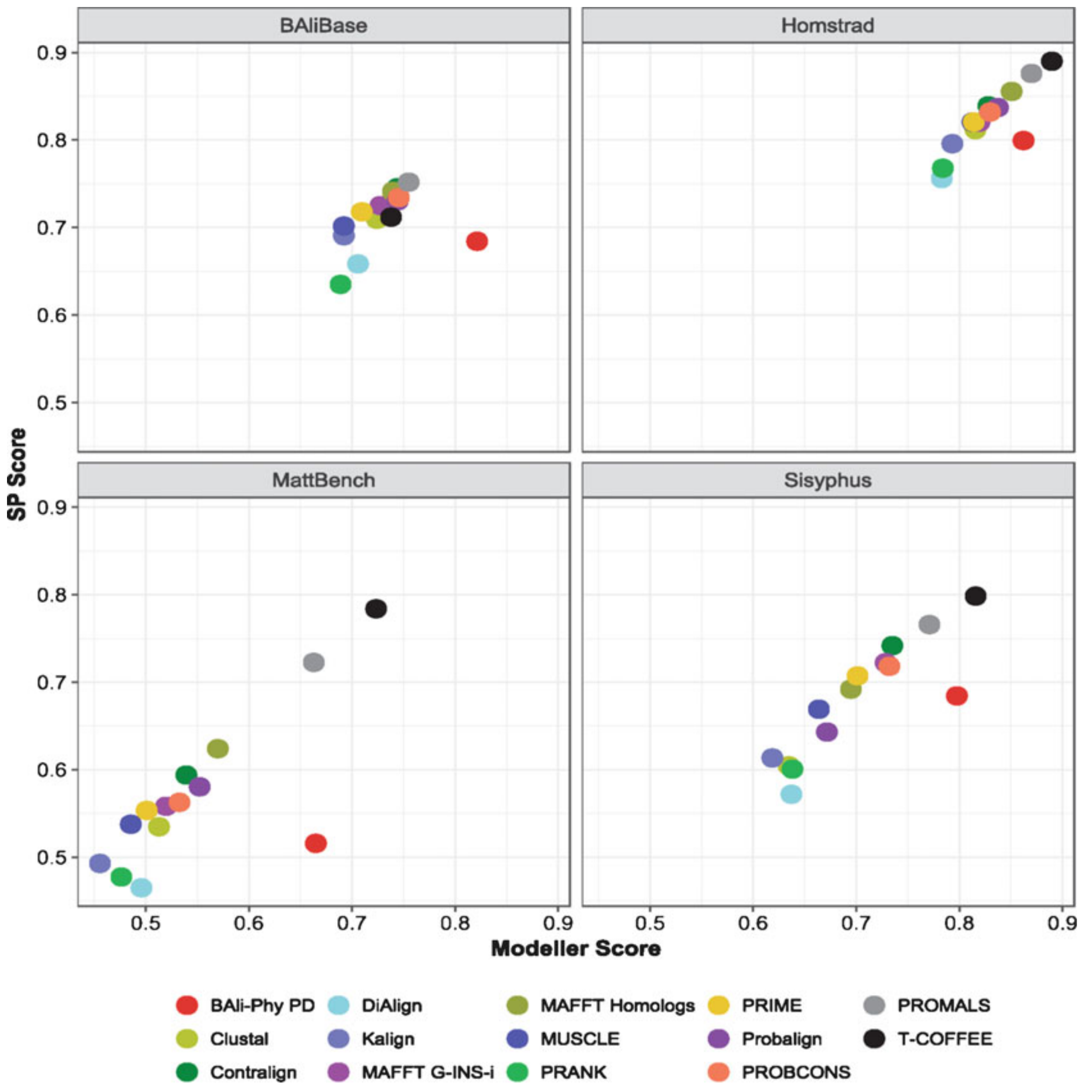


Fig. 3 [Figure 2 from [32], reprinted under the Creative Commons Attribution License (http:/creativecommons.org/licenses/by-nc/4.0/)] This figure shows SP-score and Modeler score for BAli-Phy (using the posterior decoding to produce a single alignment) and several other multiple sequence alignment methods on four protein alignment benchmarks, restricted to datasets with at most 27 sequences. Each BAli-Phy analysis for each of the datasets was allowed 48 h on 32 processors to run on the Blue Waters supercomputer at the National Center for Supercomputing Applications. Note that BAli-Phy, shown in red, has excellent Modeler score (precision) for the BAliBASE collection, and average to very good Modeler scores on the other benchmarks. However, BAli-Phy is only middling (and in some cases in the bottom third) with respect to SP-score (recall) on these benchmark datasets

3.6 Challenges in Using Simulations

Simulation studies, where sequences evolve down model trees with evolutionary processes that include substitutions and indels (and perhaps other events) provide a rigorous test of accuracy, since the true alignment and true tree are known. Yet, as we have seen, relative accuracy between methods on simulated and biological datasets can be different, and one possible explanation is that the simulation models are sufficiently unrealistic that relative performance on simulated datasets is not reflective of performance on biological data. Here we examine the kinds of differences that might be in play.

The simulation models used to evaluate alignment methods, for example, in the simulation tools ROSE [53] and INDELible [54], are generally enhancements of models used in phylogeny estimation so that they also have indels. The most complex sequence evolution commonly used for phylogeny estimation on nucleotides is the GTR+GAMMA model, which is a rooted binary tree equipped with numeric parameters that define the stochastic process; furthermore, the numeric parameters assume that there is one 4×4 substitution rate matrix that governs the entire tree [47, 55, 56]. Yet this is an unrealistic assumption, as compositional heterogeneity (indicative of changed substitution rate matrices) across the tree has been observed in many biological datasets (e.g., see discussion in [57]), which has led to the introduction of parameter-rich models, such as the Generalized Markov Model [58]. More generally, deviations from the standard model assumptions are observed to increase with evolutionary distance between the taxa. The appreciation of the impact of these violations of the model assumptions on phylogeny estimation has increased in recent years [57, 59–62]. For example, Naser-Khdour et al. [60] noted:

> "In phylogenetic inference we commonly use models of substitution which assume that sequence evolution is stationary, reversible and homogeneous (SRH). Although the use of such models is often criticized, the extent of SRH violations and their effects on phylogenetic inference of tree topologies and edge lengths are not well understood. Here, we introduce and apply the maximal matched-pairs tests of homogeneity to assess the scale and impact of SRH model violations on 3,572 partitions from 35 published phylogenetic datasets. We show that roughly one quarter of all the partitions we analysed (23.5%) reject the SRH assumptions, and that for 25% of datasets, the topologies of trees inferred from all partitions differ significantly from those inferred using the subset of partitions that do not reject the SRH assumptions. This proportion of significantly different topologies is actually even greater when evaluating trees inferred using the subset of partitions that rejects the SRH assumptions, as compared to trees inferred from all partitions. These results suggest that the extent and effects of model violation in phylogenetics may be substantial. They highlight the importance of testing for model violations and possibly excluding partitions that violate models prior to tree reconstruction. Our results also suggest that further effort in developing models that do not require SRH assumptions could lead to large improvements in the accuracy of phylogenomic inference."

Naser-Khdour et al. also observed that the degree and impact of model violations depends in part on the evolutionary distance, and is less for closely related taxa than for distantly related taxa. This would be consistent with the hypothesis that the GTR substitution rate matrix should change slowly across the tree, with large changes resulting from the accumulation of smaller changes. Thus, large-scale phylogeny estimation will most likely present larger challenges than small-scale phylogeny estimation. Unfortunately, many of the interesting questions in biology require phylogenies that span large evolutionary distances, going to the origins of birds, of plants, etc.

It seems likely that model violations have an impact on multiple sequence alignment. Recall that [32] showed that BAli-Phy had the highest accuracy of all tested methods on simulated datasets but had low to middling accuracy on biological datasets. While it is not yet known why there is this discordance between results on biological and simulated datasets, one obvious hypothesis is that biological data evolve differently from the model assumed in BAli-Phy. This is consistent with current research as described above, which provided evidence that biological sequence evolution is different from the assumed models used in phylogeny estimation, and that these differences can lead to mistakes in phylogenetic inference.

The take-home lesson from these studies is troubling: the simplifying assumptions in current models of sequence evolution may have negative consequences for accuracy in both alignments and trees. The obvious approach is to allow for increased model complexity, through the use of additional numeric parameters; however, maximum likelihood estimation under such models would be extremely computationally intensive, and there is a danger of over-fitting.

3.7 Challenges in Protein Benchmarks

Several studies have pointed out challenges in using structural benchmarks [63, 64]. For example, Chatzou et al. [64] pointed out that evaluating multiple sequence alignment methods ("MSAMs") using structural benchmarks has some challenges:

> A major milestone in the development of MSAMs has been the introduction of structure-based reference alignments that can be used to compare the relative capacities of various methods to reconstruct structurally correct alignments from sequence only. The choice of structure seems rather natural because 3D features are known to be more evolutionary resilient than the underlying sequences. On the other hand, this approach relies on the unproven rationale that structurally and evolutionary correct alignments are identical. No proof exists that this assumption may be correct, and a simple reasoning suggests it may not be the case. While there can be only one correct way of matching homologous residues–the one that perfectly reflects the unique evolutionary history of the considered sequences and matches–there can be as many structurally correct alignments as there are ways to superpose the sequences with equivalent 3D compactness. Another major potential discrepancy between structural and evolutionary alignments results from convergent evolution. Whenever such a process has shaped some

portions of a sequence data set, the resulting alignment matching convergent regions will be structurally correct and evolutionary false–and reciprocally.

To the extent that correct structural alignments are not unique, this offers one explanation for the observation made by Edgar [63] that the SABmark [65] collection of pairwise alignments is not self-consistent (the other explanation is that there are errors in the pairwise alignments). However, if there are multiple correct structural alignments, then it is difficult to use structural alignments as benchmarks, since designating one of the "correct" alignments as the reference automatically means that the other correct alignments will all be considered wrong (a point also made by Chatzou et al. [64]). Another consequence is that to the extent that structural alignments do not reflect evolutionary history, trees constructed on these structural alignments may not be accurate. However, it may be that the frequency with which structurally correct alignments disagree with the true phylogenetic alignment is sufficiently low (or, even if it occurs, the differences between the two alignments may be small enough) that this issue will not be a significant one in practice.

Some of the literature (e.g., [63]) goes so far as to suggest that the biological benchmarks may not be completely accurate, even as structurally defined alignments. One explanation offered for why the multiple sequence alignments in the structural benchmarks are not perfectly accurate is that they are often inferred rather than confirmed experimentally, and there can be two or more equally feasible structural alignments between two proteins. As an example, Edgar [63] noted that BAliBASE contains sequences without known structure, so that by definition any multiple sequence alignment can only be computationally rather than experimentally inferred.

These findings raise some significant concerns about the use of structural benchmarks in evaluating alignment methods. It may well be that large differences between methods with respect to alignment criteria on these benchmarks represent real differences in accuracy, but a small improvement in alignment accuracy on a benchmark may not represent an actual improvement (i.e., to the extent that the benchmark is flawed, the predicted alignment might in fact be more accurate, even if it has a worse accuracy score on the benchmark).

The reason this is concerning is that many of the studies that have been used to evaluate protein MSA methods have concluded that one method is better than another based on rather small differences in alignment criteria on these benchmarks. For example, Sievers et al. [27] introduced Clustal-Omega, a fast multiple sequence alignment method capable of analyzing very large datasets, and evaluated it (using column score) on several protein structural benchmark collections. They found that Clustal-Omega was fast but not among the most accurate, on these benchmarks. As an example, in describing the results on the Prefab benchmark collection, they note:

> The consistency-based programs MSAprobs, MAFFT L-INS-i, Probalign, Probcons and T-Coffee, are again the most accurate but with long run times. Clustal Omega is close to the consistency programs in accuracy but is much faster. There is then a gap to the faster progressive based programs of MUSCLE, MAFFT, Kalign, and Clustal W.

Yet, the differences between TC scores for Clustal-Omega and the best performing method (here, MSAprobs) on Prefab were small (71% TC score for Clustal-Omega and 73.7% for MSAprobs), and there was an even smaller difference (only 2.3%) between Clustal-Omega and MAFFT run in default mode. Are these differences important? Perhaps. But to the extent that Prefab is not perfectly reliable, it may well be that these differences in column score do not reflect real differences in accuracy with respect to the true multiple sequence alignment.

4 Conclusion

Multiple sequence alignment is a fundamental step in many biological studies, with ramifications for many downstream analyses. Thus, errors in an estimated multiple sequence alignment have the potential to lead to faulty inferences in downstream analyses, whether these are for phylogeny estimation, protein structure prediction, the inference of positive selection, etc. Hence, accurate estimation of multiple sequence alignment is important.

The last few decades have seen a large number of new methods developed, and a continued refinement and elaboration on existing methods, many of which have performed well on both biological and simulated datasets. We also have divide-and-conquer strategies that enable alignment methods to scale to large datasets, statistical models of sequence evolution that include insertions and deletions and so enable statistical co-estimation of alignments and trees, as well as statistical alignment methods that are phylogeny-aware. Based on these advances, one might be tempted to conclude that multiple sequence alignment is *solved*. Yet, there are many reasons to argue against this conclusion.

First, standard alignment estimation methods, by design, do not address the full degree of heterogeneity present in real biological datasets, resulting (for example) from duplications of genomic regions (including tandem repeats), rearrangements, etc. Therefore, alignment methods do not enable the full discovery of homology in such circumstances.

Second, statistical estimation methods that are based on parametric models of sequence evolution may not provide good accuracy, if the biological data evolve sufficiently differently than the model predicts. Furthermore, the recent literature does suggest that many biological datasets violate these model assumptions, and that inferences under these models may be flawed. These findings also suggest that the simulations performed under these standard models may not be as relevant to understanding alignment methods as we had hoped.

Third, studies suggest that even biological benchmarks are imperfect; hence, it may be that only large differences in accuracy on these benchmarks are indicative of important differences. Unfortunately, many of the differences in methods on biological datasets have been small, meaning that the relative ranking of methods may not be as clear as we have thought.

Fourth, despite all these difficulties in evaluating methods, studies show that many methods are highly accurate under conditions with low rates of evolution, but accuracy degrades with increases in heterogeneity. Thus, alignment estimation remains difficult for large, heterogeneous datasets.

Overall, we have come to the troubling conclusion that alignment methods are essential to biological discovery, but evaluating alignment methods is itself challenging, due to the difficulties in relying on either simulations (as currently performed) or current biological benchmarks (due to identified flaws). Furthermore, it may well be that none of the current methods have sufficient accuracy for alignment of large, heterogeneous datasets. In addition, although there have been many studies evaluating alignment methods with respect to standard criteria (e.g., TC score), much less is understood about the impact of alignment on downstream analyses, such as the inference of ancestral sequences [36], prediction of protein structure and function [66, 67], the estimation of the numeric parameters in sequence evolution models, dates at internal nodes, etc.

Thus, new research is needed to develop better benchmarks, including improved simulations that are under more realistic sequence evolution models that better reflect biological evolution, and more reliable biological benchmarks. New research is also needed to evaluate how alignment estimation impacts downstream analyses, since standard alignment criteria may not reflect accuracy for these questions. Indeed, it may also be that *new alignment methods are needed*, especially for cases where molecules evolve with rearrangements, duplications, etc. Indeed, multiple sequence alignment, far from being solved, is one of the most important and yet most challenging problems in bioinformatics.

Acknowledgements

This paper was supported by NSF grant ABI-1458652 to TW. This research was also inspired by a Program on Multiple Sequence Alignment held at IPAM (Institute for Pure & Applied Mathematics, an NSF Math Institute at UCLA) in January 2015. The author wishes to thank George Chacko, whose critical comments were helpful in improving the manuscript.

References

1. Morrison D, Ellis JT (1997) Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18S rDNAs of apicomplexa. Mol Biol Evol 14:428–441
2. Hall B (2005) Comparison of the accuracies of several phylogenetic methods using protein and DNA sequences. Mol Biol Evol 22:792–802
3. Ogden T, Rosenberg M (2006) Multiple sequence alignment accuracy and phylogenetic inference. System Biol 55(2):314–328
4. Liu K, Raghavan S, Nelesen S, Linder CR, Warnow T (2009) Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science 324 (5934):1561–1564
5. Wang L-S, Leebens-Mack J, Wall PK, Beckmann K, dePamphilis CW, Warnow T (2011) The impact of multiple protein sequence alignment on phylogenetic estimation. IEEE/ACM Trans Comput Biol Bioinform 8(4):1108–1119
6. Morrison D (2006) Multiple sequence alignment for phylogenetic purposes. Aust Syst Bot 19:479–539
7. Reeck G, deHaen C, Teller D, Doolitte R, Fitch W, Dickerson R, Chambon P, McLachlan A, Margoliash E, Jukes T, Zuckerkandl E (1987) "Homology" in proteins and nucleic acids: a terminology muddle and a way out of it Cell 50:667
8. Iantorno S, Gori K, Goldman N, Gil M, Dessimoz C (2014) Who watches the watchmen? an appraisal of benchmarks for multiple sequence alignment. In Russell D (ed) Multiple sequence alignment methods. Springer, Berlin, pp 59–73
9. Cline M, Hughey R, Karplus K (2002) Predicting reliable regions in protein sequence alignments Bioinformatics 18(2):306–314
10. Löytynoja A, Goldman N (2005) An algorithm for progressive multiple alignment of sequences with insertions Proc Natl Acad Sci 102:10557–10562
11. Holmes I (2017) Historian: accurate reconstruction of ancestral sequences and evolutionary rates. Bioinformatics 33(8):1227–1229. https://doi.org/10.1093/bioinformatics/btw791
12. Edgar RC, Sjölander K (2003) SATCHMO: sequence alignment and tree construction using hidden Markov models. Bioinformatics 19(11):1404–1411
13. Hagopian R, Davidson J, Datta R, Jarvis G, Sjölander K (2010) SATCHMO-JS: a webserver for simultaneous protein multiple sequence alignment and phylogenetic tree construction Nucl Acids Res 38 (Web Server Issue): W29–W34. PMCID: PMC2896197
14. Pei J, Grishin N (2014) Promals3D: multiple protein sequence alignment enhanced with evolutionary and three-dimensional structural information. In Russell D (ed) Multiple sequence alignment methods. Springer, Berlin
15. Redelings B, Suchard M (2005) Joint Bayesian estimation of alignment and phylogeny. Syst Biol 54(3):401–418
16. Suchard, M. A. and Redelings, B. D. (2006) BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22 (16):2047–2048
17. Novák Á, Miklós I, Lyngsoe R, Hein J (2008) StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. Bioinformatics 24:2403–2404
18. Löytynoja A, Goldman N (2008) Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. Science 320(5883):1632–1635
19. Löytynoja A, Vilella A, Goldman N (2012) Accurate extension of multiple sequence alignments using a phylogeny-aware algorithm Bioinformatics 28(13):1684–1691
20. Liu K, Warnow T, Holder MT, Nelesen SM, Yu J, Stamatakis AP, Linder CR (2012) SATé-II: very fast and accurate simultaneous

estimation of multiple sequence alignments and phylogenetic trees. Syst Biol 61(1):90–106
21. Mirarab S, Nguyen N, Wang L-S, Guo S, Kim J, Warnow T (2015) PASTA: ultra-large multiple sequence alignment of nucleotide and amino acid sequences. J Comput Biol 22:377–386
22. Nute M, Warnow T (2016) Scaling statistical multiple sequence alignment to large datasets. BMC Genomics 17(10):764
23. 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 Res 30 (14):3059–3066
24. Katoh K, Kuma K, Toh H, Miyata T (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucleic Acids Res 33(2):511–518
25. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30(4):772–780
26. Katoh K, Toh H (2008) Recent developments in the MAFFT multiple sequence alignment program. Brief Bioinform 9(4):286–298
27. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Soding J, Thompson JD, Higgins DG (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7:539
28. Nelesen S, Liu K, Zhao D, Linder CR, Warnow T (2008) The effect of the guide tree on multiple sequence alignments and subsequent phylogenetic analyses. In Pacific symposium on biocomputing 2008, vol 13. World Scientific, Singapore, pp 15–24
29. Toth A, Hausknecht A, Krisai-Greilhuber I, Papp T, Vagvolgyi C, Nagy L (2013) Iteratively refined guide trees help improving alignment and phylogenetic inference in the mushroom family *bolbitiaceae*. PLoS One 8(2):e56143
30. Edgar RC (2004) MUSCLE: a multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 32 (5):1792–1797
31. Liu Y, Schmidt B, Maskell DL (2010) MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities Bioinformatics 26 (16):1958–1964
32. Nute M, Saleh E, Warnow T (2018) Evaluating statistical multiple sequence alignment in comparison to other alignment methods on protein data sets Syst Biol 68(3):396–411
33. 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
34. Edgar RC (2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 5(113):113
35. Nguyen N, Mirarab S, Kumar K, Warnow T (2015) Ultra-large alignments using phylogeny aware profiles Genome Biol 16(124). A preliminary version appeared in the Proceedings RECOMB 2015
36. Vialle RA, Tamuri AU, Goldman N (2018) Alignment modulates ancestral sequence reconstruction accuracy. Mol Biol Evol 35 (7):1783–1797
37. Thompson J, Plewniak F, Poch O (1999) BAliBASE: a benchmark alignments database for the evaluation of multiple sequence alignment programs. Bioinformatics 15:87–88. Extended collection of benchmarks is available at http://www-bio3d-igbmc.u-strasb.fr/balibase/
38. Suchard MA, Redelings BD (2006) BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22:2047–2048
39. Redelings BD, Suchard MA (2007) Incorporating indel information into phylogeny estimation for rapidly emerging pathogens. BMC Evol Biol 7:40
40. Roshan U, Livesay DR (2006) Probalign: multiple sequence alignment using partition function posterior probabilities. Bioinformatics 22:2715–2721
41. Morrison DA, Ellis JT (1997) Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18S rDNAs of Apicomplexa. Mol Biol Evol 14(4):428–441
42. Wong KM, Suchard MP, Huelsenbeck JP (2008) Alignment uncertainty and genomic analysis Science 319(5862):473 – 476
43. Cantarel BL, Morrison HG, Pearson W (2006) Exploring the relationship between sequence similarity and accurate phylogenetic trees. Mol Biol Evol 23(11):2090–2100
44. Hall BG (2005) Comparison of the accuracies of several phylogenetic methods using protein and DNA sequences. Mol Evol Biol 22 (3):792–802
45. Roshan U, Livesay D, Chikkagoudar S (2006) Improving progressive alignment for phylogeny reconstruction using parsimonious guide-trees. In Proceedings of the IEEE 6th symposium on bioinformatics and bioengineering (BIBE'06). IEEE Computer Society Press, Washington, DC, pp 159–164

46. Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4:406–425
47. Warnow T (2018) Computational phylogenetics: an introduction to designing methods for phylogeny estimation. Cambridge University Press, Cambridge
48. Wheeler T, Kececioglu J (2007) Multiple alignment by aligning alignments. Bioinformatics 23:i559–i568
49. Cannone J, Subramanian S, Schnare M, Collett J, D'Souza L, Du Y, Feng B, Lin N, Madabusi L, Muller K, Pande N, Shang Z, Yu N, Gutell R. (2002) The Comparative RNA Web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron and other RNAs. BioMed Central Bioinform 3(15). http://www.rna.ccbb.utexas.edu
50. Varón A, Vinh L, Wheeler W (2010) POY version 4: phylogenetic analysis using dynamic homologies. Cladistics 26:72–85
51. Thompson J, Higgins D, Gibson T (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res 22:4673–4680
52. Do CB, Mahabhashyam MS, Brudno M, Batzoglou S (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. Genome Res 15(2):330–340
53. Stoye J, Evers D, Meyer F (1998) Rose: generating sequence families Bioinformatics 14 (2):157–163
54. Fletcher W, Yang Z (2009) Indelible: a flexible simulator of biological sequence evolution. Mol Biol Evol 26:1879–1888
55. Tavaré S (1986) Some probabilistic and statistical problems in the analysis of DNA sequences. In Lectures on mathematics in the life sciences, vol 17. American Mathematical Society, Providence, pp 57–86
56. Yang Z (2014) Molecular evolution: a statistical approach. Oxford University Press, Oxford
57. Jermiin LS, Ho SY, Ababneh F, Robinson J, Larkum AW (2004) The biasing effect of compositional heterogeneity on phylogenetic estimates may be underestimated. Syst Biol 53 (4):638–643
58. Steel M (1994) Recovering a tree from the leaf colourations it generates under a Markov model. Appl Math Lett 7:19–24
59. Duchêne DA, Duchêne S, Ho SY (2017) New statistical criteria detect phylogenetic bias caused by compositional heterogeneity. Mol Biol Evol 34(6):1529–1534
60. Naser-Khdour S, Minh BQ, Zhang W, Stone EA, Lanfear R (2019) The prevalence and impact of model violations in phylogenetic analysis. Genome Biol Evol. https://doi.org/10.1093/gbe/evz193
61. Crotty SM, Minh BQ, Bean NG, Holland BR, Tuke J, Jermiin LS, Haeseler AV (2019) GHOST: recovering historical signal from heterotachously evolved sequence alignments. Syst Biol 69:249–264, syz051
62. White ND, Braun MJ (2019) Extracting phylogenetic signal from phylogenomic data: higher-level relationships of the nightbirds (Strisores). Mol Phylogenet Evol 141:106611. https://doi.org/10.1016/j.ympev.2019.106611
63. Edgar RC (2010) Quality measures for protein alignment benchmarks. Nucleic Acids Res 7:2415–2153
64. Chatzou M, Magis C, Chang J-M, Kemena C, Bussotti G, Erb I, Notredame C (2016) Multiple sequence alignment modeling: methods and applications. Brief Bioinform 17 (6):1009–1023
65. Van Walle IL, Wyns L (2005) SABmark-a benchmark for sequence alignment that covers the entire known fold space. Bioinformatics 21:1267–1268
66. Sjölander K (2004) Phylogenomic inference of protein molecular function: advances and challenges. Bioinformatics 20(2):170–179
67. Baker D, Sali A (2001) Protein structure prediction and structural genomics. Science 294 (5540):93–96

INDEX