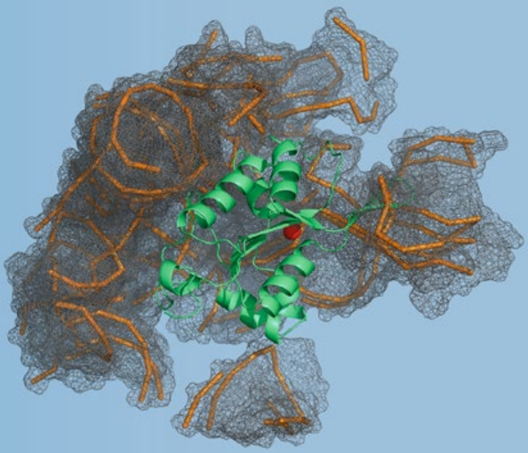
Methods in
Molecular Biology 2231

Springer Protocols



Kazutaka Katoh *Editor*

Multiple Sequence Alignment

Methods and Protocols

Humana Press

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

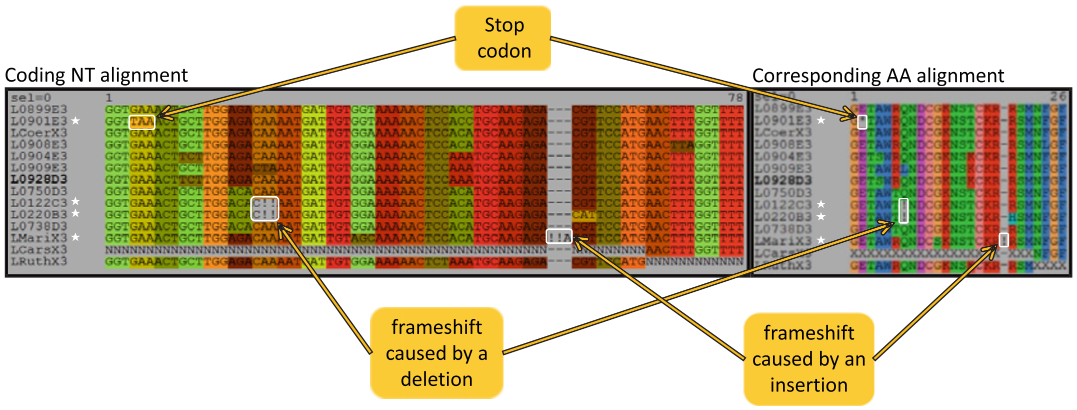

Stop codon
Coding NT alignment
Corresponding AA alignment
frameshift caused by a deletion
frameshift caused by an insertion

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

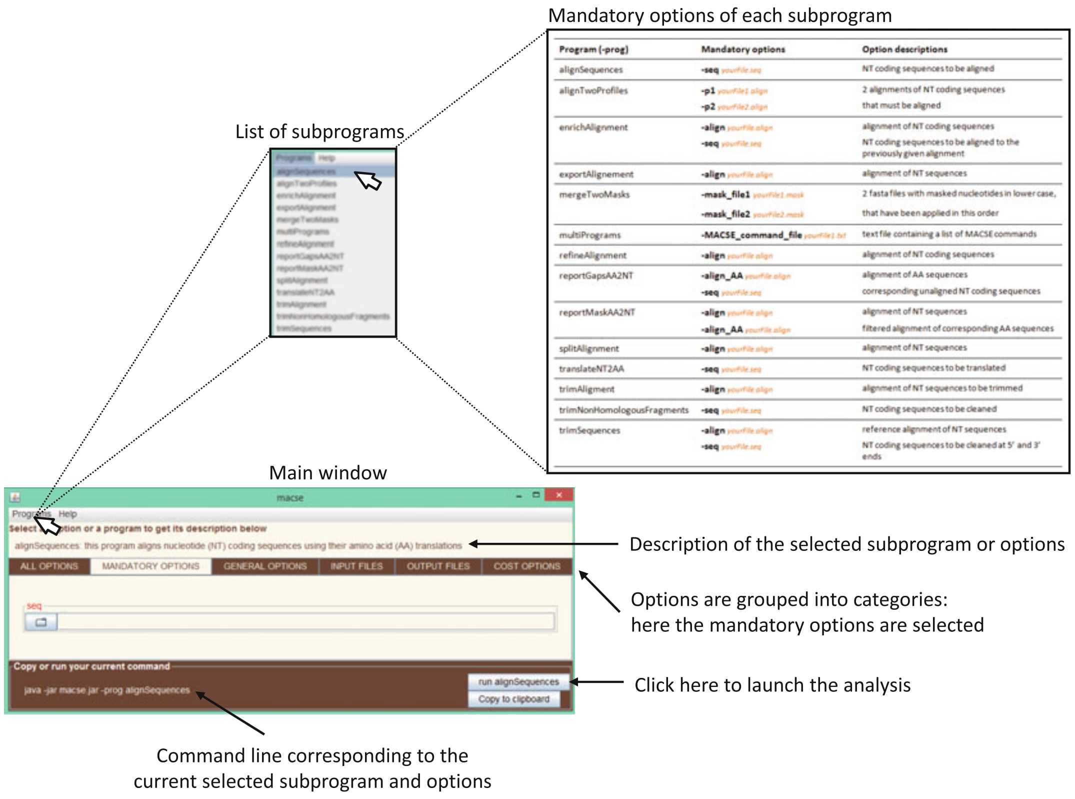
⇨ Launches the GUI version of MACSE (*see* Fig.2).

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

Copy or run your current command

```
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
```

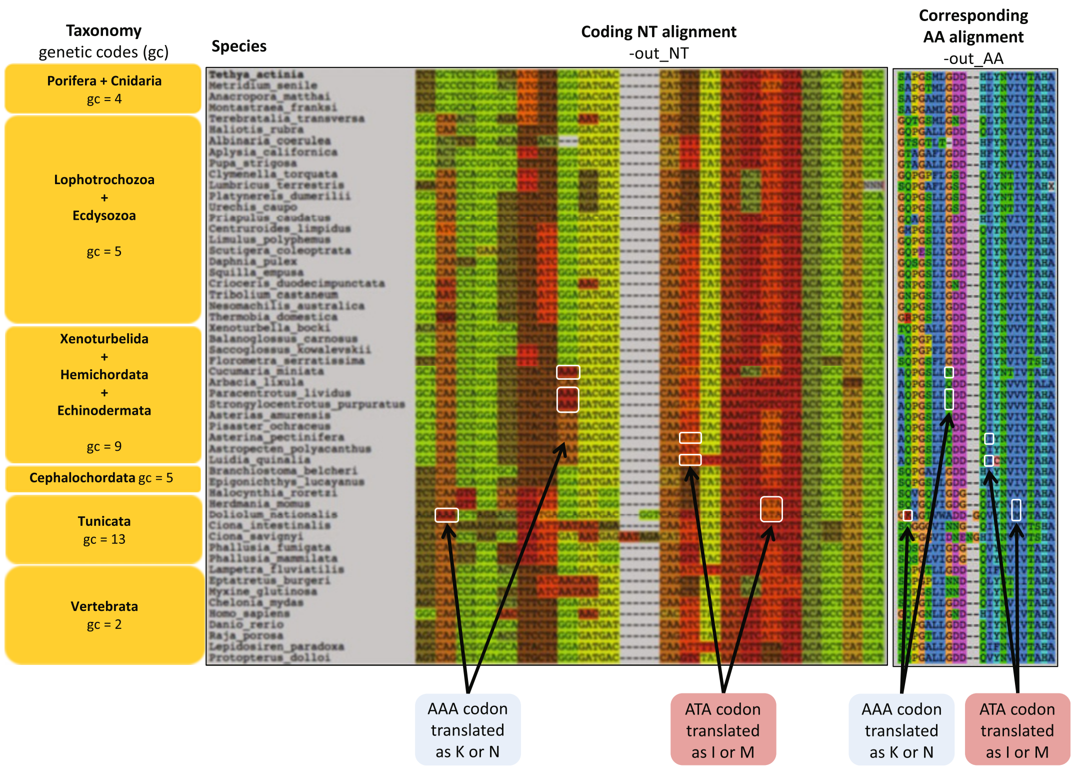


Fig. 3 MACSE alignment of 54 metazoan mitochondrial COX1 sequences from Singh et al. [14] using five different mitochondrial genetic codes corresponding to the different taxonomic groups. The nucleotide alignment (NT) and its amino acid (AA) translation are edited with SeaView ('codon-colors' option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

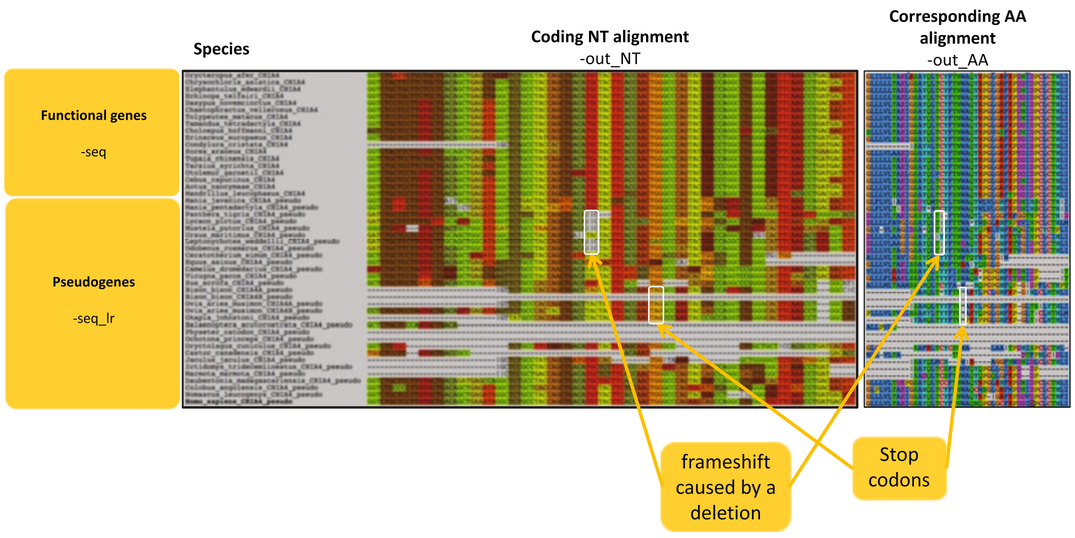


Fig. 4 MACSE alignment of 48 mammalian CHIA4 nucleotide sequences from Emerling et al. [15] containing 18 functional genes and 30 pseudogenes (_pseudo). The nucleotide alignment and its AA translation are edited with SEAVIEW ("codon-colors" option for the NT alignment). Frameshifts caused by deletions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

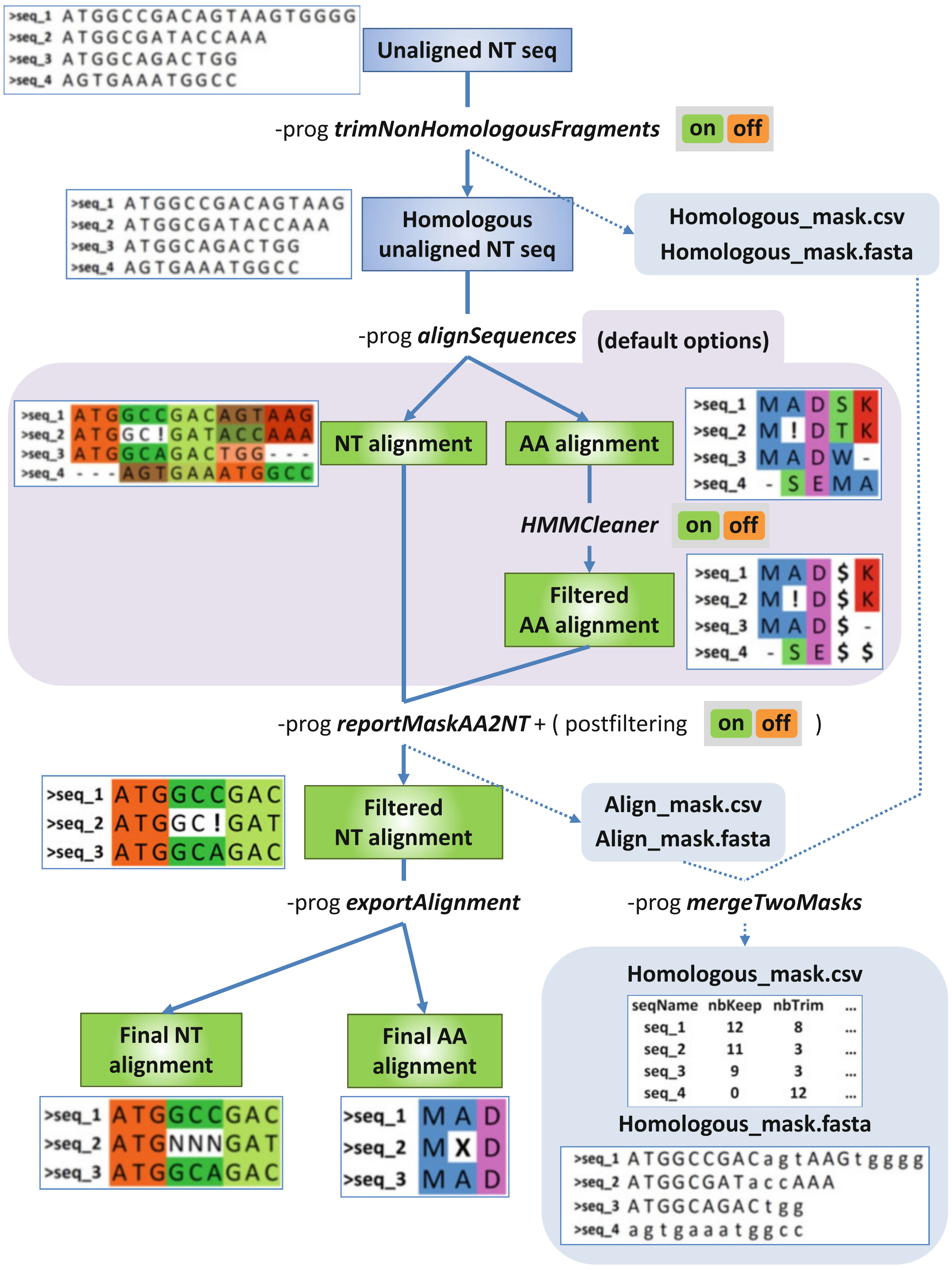
⇨ The most simple MACSE use case.

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

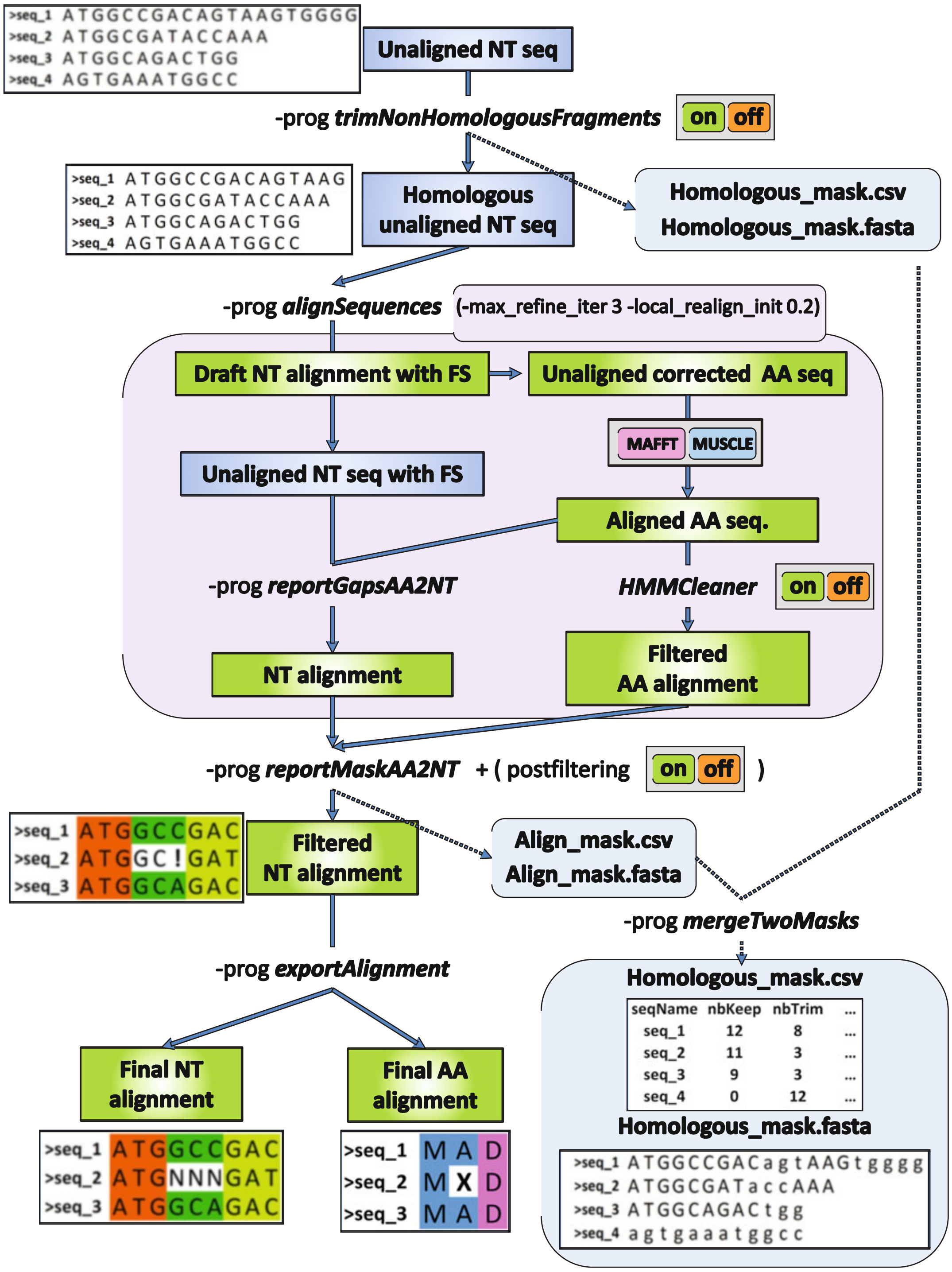
3.4 Aligning Hundreds of Sequences

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.



```
>seq_1 ATGGCCGACAGTAAGTGGGG
>seq_2 ATGGCGATACCAAA
>seq_3 ATGGCAGACTGG
>seq_4 AGTGAAATGGCC
```

```
>seq_1 ATGGCCGACAGTAAG
>seq_2 ATGGCGATACCAAA
>seq_3 ATGGCAGACTGG
>seq_4 AGTGAAATGGCC
```

```
>seq_1 ATGGCCGAC
>seq_2 ATGGC!GAT
>seq_3 ATGGCAGAC
```

```
>seq_1 ATGGCCGAC
>seq_2 ATGNNNGAT
>seq_3 ATGGCAGAC
```

```
>seq_1 MAD
>seq_2 MXD
>seq_3 MAD
```

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

```
>seq_1 ATGGCCGACagtAAGtgggg
>seq_2 ATGGCGATaccAAA
>seq_3 ATGGCAGACtgg
>seq_4 agtgaaatggcc
```

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