This is the documentation for MACSE v2.00

1. Overview

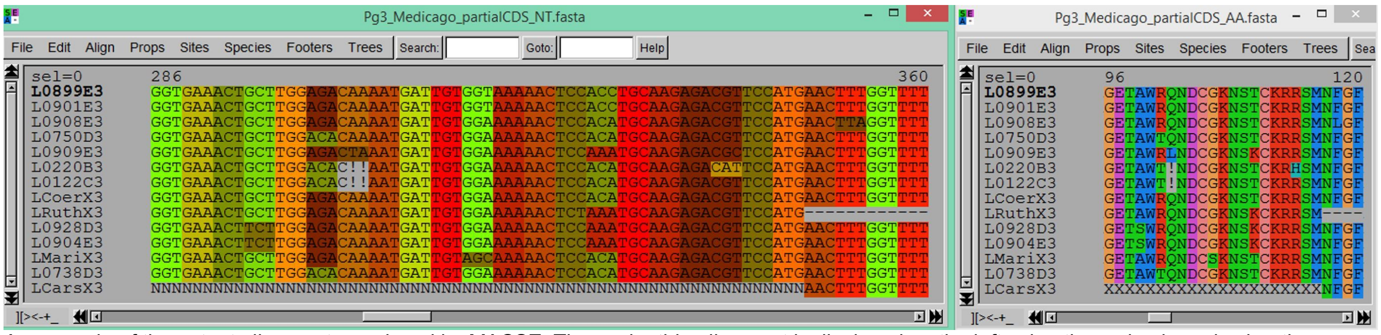
MACSE (Multiple Alignment of Coding SEquences Accounting for Frameshifts and Stop Codons) provides a complete toolkit dedicated to the multiple alignment of coding sequences that can be leveraged via both the command line and a Graphical User Interface (GUI).

Multiple sequence alignment (MSA) is a crucial step in many evolutionary analyses. Nonetheless, most existing alignment tools ignore the underlying codon structure of protein-coding nucleotide sequences. Accounting for this structure is not only useful to improve the proposed alignment, but it is also a prerequisite for some downstream analyses such as detection of selection footprints based on the ratio of non-synonymous to synonymous substitutions (dN/dS).

MACSE aligns 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. MACSE was hence the first automatic solution to align protein-coding gene datasets containing non-functional sequences (pseudogenes) without 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 a reference coding sequence.

In the output alignment produced by MACSE, frameshifts are indicated using '!'. You can specify to MACSE a subset of your sequences that are more likely to contain frameshifts or stop codons, the **less reliable sequences** in the MACSE terminology. This allows to use a lower cost when introducing a stop codon or a frameshift in those sequences as compared to introducing such events in other **reliable sequences**. More details and examples concerning this feature are provided in the alignSequences section.

Seaview is very convenient to visualize sequence alignments produced by MACSE. Indeed, Seaview accepts the '!' character in both nucleotide and amino acid sequences and also allows to visually emphasize the codon structure of the aligned nucleotide sequences.


Pg3_Medicago_partialCDS_NT.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto: Help
Pg3_Medicago_partialCDS_AA.fasta
File Edit Align Props Sites Species Footers Trees Sea
][>&lt;-+_

```
sel=0       286                                                                        360
L0899E3     GGTGAAACTGCTTGGAGACAAAATGATTGTGGTAAAAACTCCACCTGCAAGAGACGTTCCATGAACTTTGGTTTT
L0901E3     GGTGAAACTGCTTGGAGACAAAATGATTGTGGTAAAAACTCCACCTGCAAGAGACGTTCCATGAACTTTGGTTTT
L0908E3     GGTGAAACTGCTTGGAGACAAAATGATTGTGGAAAAAACTCCACATGCAAGAGACGTTCCATGAACTTAGGTTTT
L0750D3     GGTGAAACTGCTTGGACACAAAATGATTGTGGAAAAAACTCCACATGCAAGAGACGTTCCATGAACTTTGGTTTT
L0909E3     GGTGAAACTGCTTGGAGACTAAATGATTGTGGAAAAAACTCCAAATGCAAGAGACGCTCCATGAACTTTGGTTTT
L0220B3     GGTGAAACTGCTTGGACAC!!AATGATTGTGGAAAAAACTCCACATGCAAGAGACATTCCATGAACTTTGGTTTT
L0122C3     GGTGAAACTGCTTGGACAC!!AATGATTGTGGAAAAAACTCCACATGCAAGAGACGTTCCATGAACTTTGGTTTT
LCoerX3     GGTGAAACTGCTTGGAGACAAAATGATTGTGGAAAAAACTCCACATGCAAGAGACGTTCCATGAACTTTGGTTTT
LRuthX3     GGTGAAACTGCTTGGAGACAAAATGATTGTGGAAAAAACTCTAAATGCAAGAGACGTTCCATG------------
L0928D3     GGTGAAACTTCTTGGAGACAAAATGATTGTGGAAAAAACTCCAAATGCAAGAGACGTTCCATGAACTTTGGTTTT
L0904E3     GGTGAAACTTCTTGGAGACAAAATGATTGTGGAAAAAACTCCAAATGCAAGAGACGTTCCATGAACTTTGGTTTT
LMariX3     GGTGAAACTGCTTGGAGACAAAATGATTGTAGCAAAAACTCCACATGCAAGAGACGTTCCATGAACTTTGGTTTT
L0738D3     GGTGAAACTGCTTGGACACAAAATGATTGTGGAAAAAACTCCACATGCAAGAGACGTTCCATGAACTTTGGTTTT
LCarsX3     NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNAACTTTGGTTTT
```

```
sel=0       96                     120
L0899E3     GETAWRQNDCGKNSTCKRRSMNFGF
L0901E3     GETAWRQNDCGKNSTCKRRSMNFGF
L0908E3     GETAWRQNDCGKNSTCKRRSMNLGF
L0750D3     GETAWTQNDCGKNSTCKRRSMNFGF
L0909E3     GETAWRLNDCGKNSKCKRRSMNFGF
L0220B3     GETAWT!NDCGKNSTCKRHSMNFGF
L0122C3     GETAWT!NDCGKNSTCKRRSMNFGF
LCoerX3     GETAWRQNDCGKNSTCKRRSMNFGF
LRuthX3     GETAWRQNDCGKNSKCKRRSM----
L0928D3     GETSWRQNDCGKNSKCKRRSMNFGF
L0904E3     GETSWRQNDCGKNSKCKRRSMNFGF
LMariX3     GETAWRQNDCSKNSTCKRRSMNFGF
L0738D3     GETAWTQNDCGKNSTCKRRSMNFGF
LCarsX3     XXXXXXXXXXXXXXXXXXXXXNFGF
```

An example of the output alignments produced by MACSE. The nucleotide alignment is displayed on the left using the codon based coloration of SEAVIEW, the amino acid alignment is displayed on the right. Two sequences contain frameshifts.

1. Getting started with MACSE

If you are new to MACSE, you should probably start here:

- installing and running MACSE.

2. Programs

- alignSequences: aligns nucleotide (NT) coding sequences using their amino acid (AA) translations.
- alignTwoProfiles: aligns two previously computed alignments (also called profiles).
- enrichAlignment: adds sequences to a previously computed alignment.
- exportAlignment: exports alignments (different formats) and other output files.
- multiPrograms: sequentially executes multiple MACSE commands contained in a text file (one per line).
- refineAlignment: improves a previously computed alignment.
- reportGapsAA2NT: reports gaps from aligned AA sequences to the corresponding (unaligned) NT sequences.
- reportMaskAA2NT: uses a NT alignment and a filtered (masked) version of its AA translation to derive the NT alignment.
- splitAlignment: splits an alignment to extract a subset of selected sequences and/or sites.
- translateNT2AA: translates protein-coding nucleotide sequences into amino acid sequences.
- trimAlignment: trims the input alignment by removing gappy sites at the beginning/end of the alignment.
- trimNonHomologousFragments: identifies sequence fragments that do not share homology with other sequences and remove those fragments.

3. Common options

- genetic codes
- alignment costs
- compressed AA alphabet
- allowed nucleotides

MACSE installation and fundamental usage

1. Program installation and running

MACSE is really easy to install. Just download the jar file of the latest release of MACSE (e.g. macse*v2.0.jar) on your computer. Note that JAVA (JRE above 1.5) should be installed on your computer (*download JAVA *).*

To launch the graphical version of MACSE (GUI), double click on the jar file, or run the command line without any option:

- *java -jar macse.jar **

If you want to run the command line version of MACSE you need to open a console.

- **Mac OS X:** Double-click on the *terminal* program that is in the Application/Utilities folder.
- **WINDOWS:** Get the windows invite by typing Windows + r. A small dialog windows should appear, then type *cmd* and click OK.
- **LINUX:** if you are using Linux OS you surely know how to get a console.

2. Using the graphical user interface (GUI) of MACSE

The main menu of the graphical interface lets you select one of the MACSE subprograms. Once the desired subprogram is selected, its options are displayed and can be set as you wish. As some subprograms have many options, they are sorted out in different groups/tabs:

- One tab contains only the mandatory options (indicated in red in other tabs)
- One tab contains all available options
- Other tabs contain a subset of options grouped by theme (output parameters, cost parameters, input files, etc...)

Once you have set options to the desired values, you can launch the task execution by clicking on the run button (bottom right).

Note that the command line corresponding to the options you have chosen is automatically created and displayed on the bottom left part of the window. You can hence use the GUI to easily generate the command line you need on a single example, copy this command (using copy/paste or the **copy to clipboard** button) and adapt this command to run it on multiple datasets using the command line facilities of MACSE.

Note also that every time you select a subprogram or click on an option field, a **brief help** is displayed on the top part of the window.

3. Using the command line version of MACSE

MACSE is a toolkit made of several (sub)programs. Options with the same name have identical roles in the different subprograms but each subprogram also have its own specific options. The first thing when using MACSE is to indicate the program you want to use thanks to the **-prog** option. If you have no idea, just put a random program name and MACSE will remind you the list of valid program names followed by a one line description:

- *java -jar macse.jar -prog wrongProgram*

As the **wrongProgram** parameter is not a valid program, MACSE will display the list of valid program names (be careful MACSE is case sensitive).

Note that if you attempt to run MACSE without any option, it will not display any help but instead launch its graphical mode.

Note also that all documentation examples are provided using the convention that your jar file for MACSE is **macse.jar** whereas you probably have something like **macse_v2.1.jar**.

Once you have chosen a program, you can list all its available options with the following command:

- *java -jar macse.jar -prog splitAlignment*

This example will list all **splitAlignment** program options.

Note that the options are also displayed if incorrect options are used or if mandatory ones are missing.

If you have **numerous long sequences** MACSE can need more memory than provided by the default java parameter. You can specify that you want to allocate more memory to MACSE using the **Xmx** option of java. For instance **java -jar -Xmx600m macse.jar** will allocate 600 mega of memory to your java program.

Sequence alignment

The MACSE subprogram **alignSequences** aligns protein-coding sequences at the nucleotide level while scoring the considered nucleotide alignments based on their amino acid translation. It thus favours nucleotide gap stretches that are multiple of three but also considers those inducing frameshifts, when they allow to recover the underlying codon structure. MACSE therefore produces alignments which benefit from the higher similarity of amino acid sequences while accounting for frameshifts and stop codons that could occur in pseudogenes or in poor quality sequences (alignment costs).

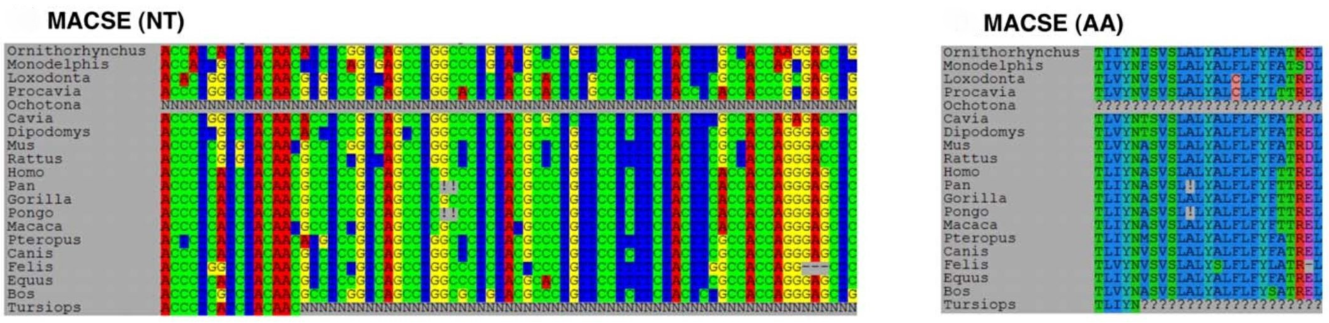
1. Basic usage

Folder: samples/alignSequences/

The **alignSequences** program aligns protein-coding nucleotide sequences provided in a FASTA file. Here is a basic example of its usage on a subset of the TMEM184 CDS provided in the release 59 of the EnsEMBL database:

- *java -jar macse.jar -prog alignSequences -seq TMEM184_Ensembl_small.fasta*

In this example, MACSE aligns nucleotide sequences found in the *TMEM184_Ensembl_small.fasta* file using MACSE default parameters. The resulting nucleotide alignment is saved in *TMEM184_Ensembl_small_NT.fasta* and the corresponding amino acid alignment is saved in *TMEM184_Ensembl_small_AA.fasta*. Note that, in this example, using MACSE reveals undocumented frameshifts in Pan and Pongo sequences (traceable by the character '!'). Those 'frameshifts' are probably due to annotation errors, more details in Ranwez et al 2011, Fig 4.



```
Ornithorhynchus  TIIYNISVSLALYALFLFYFATKEL
Monodelphis      TIVYNFSVSLALYALFLFYFATSDL
Loxodonta        TLVYNVSVSLALYALCLFYFATREL
Procavia         TLVYNVSVSLALYALCLFYLTTREL
Ochotona         ?????????????????????????
Cavia            TLVYNTSVSLALYALFLFYFATRDL
Dipodomys        TLVYNTSVSLALYALFLFYFATREL
Mus              TLVYNASVSLALYALFLFYFATRDL
Rattus           TLVYNASVSLALYALFLFYFATRDL
Homo             TLIYNASVSLALYALFLFYFTTREL
Pan              TLIYNASVSL!LYALFLFYFTTREL
Gorilla          TLIYNASVSLALYALFLFYFTTREL
Pongo            TLIYNASVSL!LYALFLFYFTTREL
Macaca           TLIYNASVSLALYALFLFYFTTREL
Pteropus         TLIYNMSVSLALYALFLFYFATREL
Canis            TLIYNVSVSLALYALFLFYFATREL
Felis            TLVYNVSVSLALYSLFLFYLATR-L
Equus            TLIYNVSVSLALYALFLFYFATREL
Bos              TLVYNASVSLALYALFLFYSATREL
Tursiops         TLIYN????????????????????
```

Figure 4. Alignments of orthologous CDS of the *tmem184a* gene (ENSG00000215155) from EnsEMBL v59.

The **seq** option is the only mandatory option for this MACSE subprogram. Any gaps present in the input FASTA file will be ignored, if you want to improve an existing alignment you should instead use the **refineAlignment** subprogram.

2. Changing default output file names

Folder: samples/alignSequences/outputs/

The names of the fasta alignment files created by MACSE can be specified using **out_NT** and **out_AA** options as follow:

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -out_NT output_NT.fasta -out_AA output_AA.fasta*

3. Less reliable sequences

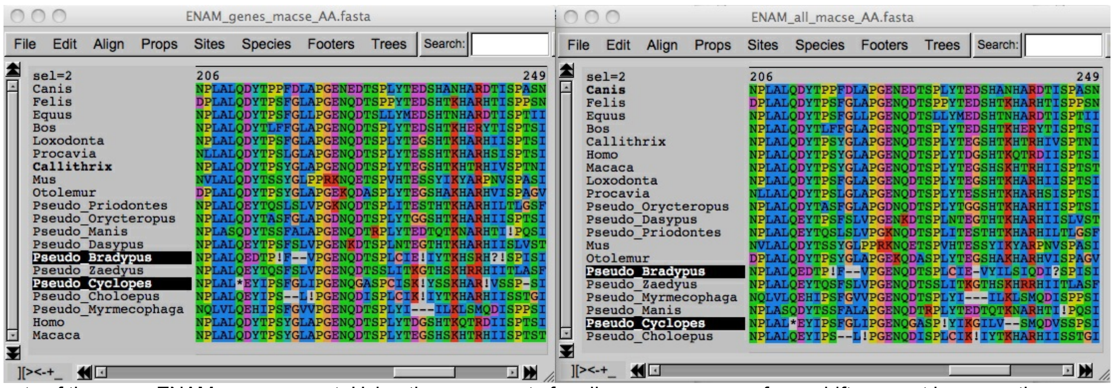
Folder: samples/alignSequences/seq_lr/

In case you think some sequences are more likely to contain frameshifts and/or stop codons than others, it is probably better to use different frameshift and stop codon costs for those **less reliable (lr)** sequences. MACSE allows you to do treat those less reliable sequences differently by simply putting them in a different fasta file, and using the **seq_lr** option to indicate the name of the fasta file containing them:

- *java -jar macse.jar -prog alignSequences -seq AMBN_coding.fasta -seq_lr AMBN_pseudo.fasta*

In this example, we suspect that some AMBN sequences involved in tooth enamel structure are pseudogenes in toothless species. We thus split our sequences in two files *AMBN_coding.fasta* and *AMBN_pseudo_lr.fasta*, so that MACSE could align all those sequences together using a smaller penalty for inserting frameshifts and stop codons in the pseudogene sequences.

Differentiating reliable from less reliable sequences is especially relevant when your dataset contains, for instance, reference sequences from model organisms found in public database and newly annotated sequences or pseudogenes.



```
sel=2                 206                                       249
Canis                 NPLALQDYTPPFDLAPGENEDTSPLYTEDSHANHARDTISPASN
Felis                 DPLALQDYTPSFGLAPGENQDTSPPYTEDSHTKHARHTISPPSN
Equus                 NPLALQDYTPSFGLLPGENQDTSLLYMEDSHTNHARDTISPTII
Bos                   NPLALQDYTLFFGLAPGENQDTSPLYTEDSHTKHERYTISPTSI
Loxodonta             NPLALQDYTPSFGLAPGENQDTSPLYTEGSHTKHARHIISPTSI
Procavia              NLLALQDYTPSLGLAPGENQDTSPLYTESSHTKHARHSISPTSI
Callithrix            NPLALQDYTPSYGLAPGENQDTSPLYTEGSHTKHTRHIVSPTNI
Mus                   NVLALQDYTSSYGLPPRKNQETSPVHTESSYIKYARPNVSPASI
Otolemur              DPLALQDYTPSYGLAPGEKQDASPLYTEGSHAKHARHVISPAGV
Pseudo_Priodontes     NPLALQEYTQSLSLVPGKNQDTSPLITESTHTKHARHILTLGSF
Pseudo_Orycteropus    NPLALQDYTASFGLAPGDNQDTSPLYTGGSHTKHARHIISPTSI
Pseudo_Manis          NPLASQDYTSSFALAPGENQDTRPLYTEDTQTKNARHTI!PQSI
Pseudo_Dasypus        NPLALQEYTPSFSLVPGENKDTSPLNTEGTHTKHARHIISLVST
Pseudo_Bradypus       NPLALQEDTP!F--VPGENQDTSPLCIE!IYTKHSRH?!SPISI
Pseudo_Zaedyus        NPLALQEYTQSFSLVPGENQDTSSLITKGTHSKHRRHIITLASF
Pseudo_Cyclopes       NPLAL*EYIPSFGLIPGENQGASPCISK!YSSKHAR!VSSP-SI
Pseudo_Choloepus      NPLALQEYIPS--L!PGENQDISPLCIK!IYTKHARHIISSTGI
Pseudo_Myrmecophaga   NQLVLQEHIPSFGVVPGENQDTSPLYI---ILKLSMQDISPPSI
Homo                  NPLALQDYTPSYGLAPGENQDTSPLYTDGSHTKQTRDIISPTSI
Macaca                NPLALQDYTPSYGLAPGENQDTSPLYTEGSHSKHTRHIISPTST
```

```
sel=2                 206                                       249
Canis                 NPLALQDYTPPFDLAPGENEDTSPLYTEDSHANHARDTISPASN
Felis                 DPLALQDYTPSFGLAPGENQDTSPPYTEDSHTKHARHTISPPSN
Equus                 NPLALQDYTPSFGLLPGENQDTSLLYMEDSHTNHARDTISPTII
Bos                   NPLALQDYTLFFGLAPGENQDTSPLYTEDSHTKHERYTISPTSI
Callithrix            NPLALQDYTPSYGLAPGENQDTSPLYTEGSHTKHTRHIVSPTNI
Homo                  NPLALQDYTPSYGLAPGENQDTSPLYTDGSHTKQTRDIISPTSI
Macaca                NPLALQDYTPSYGLAPGENQDTSPLYTEGSHSKHTRHIISPTST
Loxodonta             NPLALQDYTPSFGLAPGENQDTSPLYTEGSHTKHARHIISPTSI
Procavia              NLLALQDYTPSLGLAPGENQDTSPLYTESSHTKHARHSISPTSI
Pseudo_Orycteropus    NPLALQDYTASFGLAPGDNQDTSPLYTGGSHTKHARHIISPTSI
Pseudo_Dasypus        NPLALQEYTPSFSLVPGENKDTSPLNTEGTHTKHARHIISLVST
Pseudo_Priodontes     NPLALQEYTQSLSLVPGKNQDTSPLITESTHTKHARHILTLGSF
Mus                   NVLALQDYTSSYGLPPRKNQETSPVHTESSYIKYARPNVSPASI
Otolemur              DPLALQDYTPSYGLAPGEKQDASPLYTEGSHAKHARHVISPAGV
Pseudo_Bradypus       NPLALQEDTP!F--VPGENQDTSPLCIE-VYILSIQDI?SPISI
Pseudo_Zaedyus        NPLALQEYTQSFSLVPGENQDTSSLITKGTHSKHRRHIITLASF
Pseudo_Myrmecophaga   NQLVLQEHIPSFGVVPGENQDTSPLYI---ILKLSMQDISPPSI
Pseudo_Manis          NPLASQDYTSSFALAPGENQDTRPLYTEDTQTKNARHTI!PQSI
Pseudo_Cyclopes       NPLAL*EYIPSFGLIPGENQGASP!YIKGILV--SMQDVSSPSI
Pseudo_Choloepus      NPLALQEYIPS--L!PGENQDISPLCIK!IYTKHARHIISSTGI
```

Two alignments of the same ENAM sequence set. Using the same costs for all sequences some frameshifts cannot be correctly recovered in Bradypus and Cyclopes pseudogenes, image on the right. Using lower frameshift and stop codon penalties for known pseudogenes allows to produce a better alignment, image on the left.

If you just suspect pseudogenization, you can use a two step approach. In a first step, you can run alignSequences with a single fasta file (and faster optimization, see below) and use the first draft alignment obtained to identify possible pseudogene sequences. Then, in a second step, you can re-launch alignSequences using the option **seq_lr** with a fasta file containing those identified putative pseudogenes.

4. Controlling degree of optimization and speed

Folder: samples/alignSequences/maxRefines/

You should be aware that MACSE could be time consuming. If you just want a rapid first draft of your alignment (e.g. to check if your sequences potentially have frameshifts) or if you know that your sequences are easy to align since highly similar, you can speed up the alignment process.

Once having a first multiple sequence alignment (MSA), MACSE uses a classical 2-cut refinement strategy to improve it. This strategy consists in partitioning the current alignment into two sub-alignments that are subsequently re-aligned. The resulting MSA replaces the previous one if its score is improved. This 2-cut refinement strategy also uses the guide tree: it iteratively considers each clade of the guide tree and splits the current global alignment so that one of the two sub-alignments contains the exact sequences of the concerned clade. Once all (external) branches have been tested, if the score has been improved, the guide tree is updated and a new iteration of this **refinement loop** will start.

Note that once you have an alignment, if you split it in two sub-alignments you can keep track of the relative position of each sites of the first alignment compared to the second one. By default MACSE uses this information to go a little bit faster by limiting its search.

Four options allow to control the balance between optimization speed and alignment accuracy:

- **optim** allows you to control whether you use standart 2-cut (default), only remove and realign one sequence at a time (*-optim 1*) or only want to know the score of your alignment (*-optim 0* no refinement at all).

- **max_refine_iter** allows you to define the maximum number of refinement iterations, i.e. how many time the split/re-alignments induced by the guide tree (external) branches will be tested in the worst case.

- **local_realign_init** if smaller than 1 the first refinement loop will consider only local improvement (the lower this value the faster the initial refinement, possible values [0-1]).

- **local_realign_dec** if smaller than 1 each refinement loop will be faster than the previous one by focusing on a smaller interval during alignment improvement steps (the lower this value the faster the refinements, possible values [0-1])

For instance to get the first draft alignment done by MACSE, prior to any optimization, use this command line:

- *java -jar macse.jar -prog alignSequences -seq TMEM184_Ensembl_Plos.fasta -max_refine_iter 0*

If you want a relatively quick optimization, you can try for instance:

- *java -jar macse.jar -prog alignSequences -seq TMEM184_Ensembl_Plos.fasta -max_refine_iter 3 -local_realign_init 0.3 -local_realign_dec 0.2*

If you want an extensive search equivalent to what was done in MACSE v1 use:

- *java -jar macse.jar -prog alignSequences -seq TMEM184_Ensembl_small.fasta -local_realign_init 1 -local_realign_dec 1*

5. Other options (genetic codes, alignment costs, AA alphabets, etc...)

You can find other options related to this program from the following links:

- genetic codes
- alignment costs
- compressed AA alphabet
- allowed nucleotides
- refineAlignment MACSE subprogram to improve an existing alignment(./refineAlignment.html)

Aligning or merging two alignments

A key step in MACSE optimization procedure is aligning two multiple sequence alignments (also called profiles), each of which containing a subset of the input sequences. This procedure is the same as 2-cut strategy used in muscle and in many other multiple sequence alignment software.

If you have already two nucleotide alignments/profiles (p1.fasta and p2.fasta) containng homologous sequences you can align them with the subprogram **alignTwoProfiles** of MACSE to produce a unique alignment p1_p2.fasta containing all the sequences. Note that **alignTwoProfiles** cannot modify the input profiles as it will only insert gaps (or frameshifts) into them. In other words, any pair of nucleotides that was in the same site/column in p1.fasta (or in p2.fasta) will still be in the same site/column in p1_p2.fasta.

1. Basic usage

This program can be seen as one step of the 2-cut optimisation done by **alignSequences** or **refineAlignment**, it thus could be helpful to look at alignSequences help pages before using it.

Folder: samples/alignTwoProfiles/

The same options as those used for alignSequences allow you to (1) control for the balance between optimization and speed (**-optim**, **-max_refine_iter**, **local_realign_init**, **local_realign_dec**), (2) specify a subset of less reliable sequences with different costs for their frameshifts and stop codons (**-seq_lr**), (3) choose the name of output files (**out_NT**, **out_AA**) and (4) select your own elementary alignment costs:

- *java -jar macse.jar -prog alignTwoProfiles -p1 p1.fasta -p2 p2.fasta*
- *java -jar macse.jar -prog alignTwoProfiles -p1 p1.fasta -p2 p2.fasta -out_NT output_NT.fasta -out_AA output_AA.fasta*
- *java -jar macse.jar -prog alignTwoProfiles -p1 p1.fasta -p2 p2.fasta -seq sequences.fasta -seq_lr sequences_lr.fasta*

2. Possible strategy for aligning a very large number of sequences

This program could be useful to build an alignment with a very large number of sequences. You can start by clustering them (either based on homology or taxonomy), then aligning independently each cluster of sequences and then merging those alignments using **alignTwoProfiles** as many times as required. A basic alignment refinement (e.g. **-optim -1**) of the global alignment can then be used to try to improve the final result.

Alternatively, you can produce a codon consensus sequence for each aligned cluster (exportAlignment), align those consensus sequences, and use the resulting alignment to produce the global alignment by reporting gaps of each consensus sequence into the corresponding cluster alignment (reportGapNT2AA).

3. Related documentation

You can find other options related to this program from the following links:

- genetic codes
- alignment costs
- compressed AA alphabet
- allowed nucleotides
- MACSE subprogram to align sequences
- MACSE subprogram to refine an existing alignment

Adding sequences to a previously computed alignment

Folder: samples/enrichAlignment/

If you have previously computed a protein-coding nucleotide alignment respecting the reading frame, you can use the **enrichAlignment** subprogram to (conditionally) add new sequences to this alignment. The options are the same as those existing for alignSequences. You can (1) specify a subset of less reliable sequences for which you can define different costs for their frameshifts and stop codons (**-seq_lr**), (2) choose the name of the output files (**out_NT**, **out_AA**), (3) specify the adequate genetic code (**gc_def**), or (4) select your own elementary alignment costs alignment costs.

1. Basic usage

The basic usage with a single type of sequences is:

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta*

and if you need to specify that some sequences are less reliable (e.g. newly sequenced, or pseudogenes):

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -seq_lr sequences_lr.fasta*

By default **enrichAlignment** adds sequences to an alignment (referred as the initial alignment) in a sequential mode: each sequence is aligned with the current alignment that contains the sequences of the initial alignment plus those previously added. Some options of **enrichAlignment** allow you to specify sequences that should be discarded and/or that all the new sequences should be aligned with the unmodified initial alignment (see below).

2. Selecting the sequences to add and those to discard

For each sequence to add, the number of stop codons, frameshifts, codon insertions and codon deletions that will be needed to align this sequence with the current alignment is counted and stored in a tabular file (see below for field details). Those counts may be used to define criteria that the additional sequences should fulfill to be actually kept in the final alignment. For instance, sequences that, once aligned, contain too many stop codons (e.g. > 0), too many deletions (e.g. > 5), too many frameshifts (e.g > 2) or too many internal insertions (e.g >0) could be automatically discarded, using the following options:

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -maxSTOP_inSeq 0*
- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -maxDEL_inSeq 5*
- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -maxFS_inSeq 2*
- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -maxINS_inSeq 0*

In the latter example, any new sequence with at least one codon insertion will be discarded. Note that this option only counts *internal gaps* insertions, i.e. those that will required to insert gaps in the middle of the current alignment. If the tested sequence only has longer 3' or 5' ends it will be kept.

The total number of codon insertions (internal or not) can also be bounded using:

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -maxTotalINS_inSeq 1*

All these event upper bound options accept any positive integer as parameters and can be combined to provide a relatively fine tuning of the sequences to be added:

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -maxSTOP_inSeq 0 -maxDEL_inSeq 5 -maxFS_inSeq 2 -maxINS_inSeq 0*

WARNING

This criteria is not sufficient to ensure that a sequence is actually homologous to those within the input alignment. Verifying that sequences are homologous and can be added to the alignment must be done before using **enrichAlignment**, for instance using usearch/uclust programs *or* trimNonHomologousFragments *(a subprogram of MACSE developed for this purpose). Indeed, a short sequence of few amino acids will, in most cases, be aligned without the need to introduce frameshifts or stop codons and will not be eliminated. Conversely an homologous sequence may contain frameshifts or stop codons.*

3. Fixed alignment and parallelization

Working with a fixed alignment is especially convenient when dealing with (meta)barcoding data since this usually requires to deal with numerous highly similar sequences that are not expected to have indels. When using this option, all sequences to be added are compared with the same initial alignment. The key advantage is that this allows to parallelize the task. For example, if you have 50,000 contigs/sequences that you want to add to your initial alignment, you can split this large dataset in 50 sets of 1,000 sequences each and run the task on 50 computers/CPUs. Moreover, if each of your 50,000 sequences can be correctly aligned with the original alignment without inserting gap events in this original alignment, then the aligned version of your 50,000 sequences (that were computed independently) can be merged to your initial alignment to get a valid global alignment. To ensure this property, we automatically set **maxINS_inSeq=0** to identify the subset of the sequences that matches this condition (i.e. no insertion allowed).

- **Allowing two different kinds of frameshifts with a fixed alignment**

A sequence to add can have two kinds of frameshifts: those where 1 or 2 nucleotides are missing and those where 1 or 2 extra nucleotides are present. The first ones does not impede a sequence to be added to a fixed alignment, whereas the second does. To allow the insertion of sequences having just one frameshift with 1 or 2 nucleotides, those 1 or 2 nucleotides could simply be deleted from the sequences. This is not done by default (as alignment normally do not modify input sequences) but you can allow it using the following options:

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -fixed_alignment_ON -new_seq_alterable_ON*

- **Allowing to trim a few nucleotides at the extremities of the sequences**

Sequence extremities are often of lower qualit and this can lead to insertion/deletion near sequences extremities. You can allow MACSE to trim a few nucleotides at the sequence extremities to avoid those insertion/deletion events by specifying the maximal number of nucleotide that can be trimed at each end of the sequences:

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -fixed_alignment_ON -max_NT_trimmed 1*

- **Concatenation of aligned sequences**

If you have split the sequences you want to add into smaller subsets, and used **enrichAlignment** to add them (independently) to your initial alignment, you will need afterwards, to concatenate all your aligned sequences. This will be much easier if each output file only contains the added sequences of the corresponding subset (and not also the initial alignment that would be, by default, in each output file of enrichAlignment), as allowed by the option:

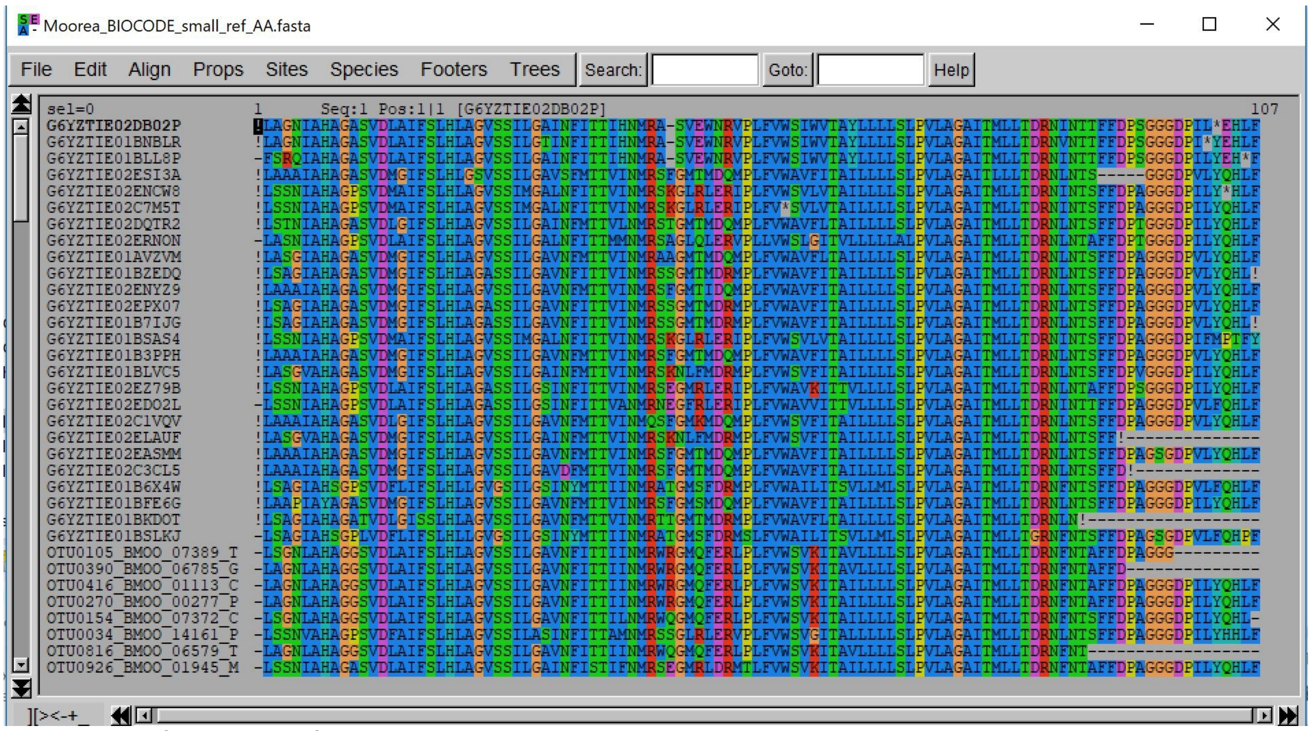
- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -fixed_alignment_ON -output_only_added_seq_ON*

4. A realistic example related to metabarcoding

Metabarcoding analysis often requires to handle thousands of sequences. Such datasets are not directly tractable with the **alignSequence** subprogram of MACSE, but they can be handled by sequentially adding your newly obtained sequences to a reference alignment containing sequences of related taxa for your targeted locus (COX1, matK, rbcL, etc...). We successfully used this approach in the Moorea project, M. Leray et al 2013. As we had a initial alignment, used in previous work, we started by using refineAlignment to identify some potential errors in this alignment.

The sequences from the initial alignment are grouped and are given a name starting with *OTU* whereas added sequences have a name starting with *G6YZT*. Here the analysis is done on a small subset of the original alignment, using rather strict conditions for insertion and default incremental insertion:

- *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*



```
sel=0                 1        Seq:1 Pos:1|1 [G6YZTIE02DB02P]                                              107
G6YZTIE02DB02P      !LAGNIAHAGASVDLAIFSLHLAGVSSILGAINFITTIHNMRA-SVEWNRVPLFVWSIWVTAYLLLLSLPVLAGAITMLLTDRNINTTFFDPSGGGDPIL*EHLF
G6YZTIE01BNBLR      !LAGNIAHAGASVDLAIFSLHLAGVSSILGTINFITTIHNMRA-SVEWNRVPLFVWSIWVTAYLLLLSLPVLAGAITMLLTDRNVNTTFFDPSGGGDPI*YEHLF
G6YZTIE01BLL8P      -FSRQIAHAGASVDLAIFSLHLAGVSSILGAINFITTIHNMRA-SVEWNRVPLFVWSIWVTAYLLLLSLPVLAGAITMLLTDRNINTTFFDPSGGGDPILYEH*F
G6YZTIE02ESI3A      !LAAAIAHAGASVDMGIFSLHLGSVSSILGAVSFMTTVINMRSFGMTMDQMPLFVWAVFITAILLLLSLPVLAGAITLLLTDRNLNTS-----GGGDPVLYQHLF
G6YZTIE02ENCW8      !LSSNIAHAGPSVDMAIFSLHLAGVSSIMGALNFITTVINMRSKGLRLERIPLFVWSVLVTAILLLLSLPVLAGAITMLLTDRNINTSFFDPAGGGDPILY*HLF
G6YZTIE02C7M5T      !LSSNIAHAGPSVDMAIFSLHLAGVSSIMGALNFITTVINMRSKGLRLERIPLFV*SVLVTAILLLLSLPVLAGAITMLLTDRNINTSFFDPAGGGDPILYQHLF
G6YZTIE02DQTR2      !LSTNIAHAGASVDLGIFSLHLAGVSSILGAINFMTTVLNMRSTGMTMDQMPLFVWAVFLTAILLLLSLPVLAGAITMLLTDRNLNTSFFDPTGGGDPILYQHLF
G6YZTIE02ERNON      -LASNIAHAGPSVDLAIFSLHLAGVSSILGALNFITTMMNMRSAGLQLERVPLLVWSLGITVLLLLLALPVLAGAITMLLTDRNLNTAFFDPTGGGDPILYQHLF
G6YZTIE01AVZVM      !LASGIAHAGASVDMGIFSLHLAGVSSILGAVNFMTTVINMRAAGMTMDQMPLFVWAVFLTAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHLF
G6YZTIE01BZEDQ      !LSAGIAHAGASVDMGIFSLHLAGASSILGAVNFITTVINMRSSGMTMDRMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHL!
G6YZTIE02ENYZ9      !LAAAIAHAGASVDMGIFSLHLAGVSSILGAVNFMTTVINMRSFGMTIDQMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHLF
G6YZTIE02EPX07      !LSAGIAHAGASVDMGIFSLHLAGASSILGAVNFITTVINMRSSGMTMDRMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHLF
G6YZTIE01B7IJG      !LSAGIAHAGASVDMGIFSLHLAGASSILGAVNFITTVINMRSSGMTMDRMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHL!
G6YZTIE01BSAS4      !LSSNIAHAGPSVDMAIFSLHLAGVSSIMGALNFITTVINMRSKGLRLERIPLFVWSVLVTAILLLLSLPVLAGAITMLLTDRNINTSFFDPAGGGDPIFMPTFY
G6YZTIE01B3PPH      !LAAAIAHAGASVDMGIFSLHLAGVSSILGAVNFMTTVINMRSFGMTMDQMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHLF
G6YZTIE01BLVC5      !LASGVAHAGASVDMGIFSLHLAGVSSILGAINFMTTVINMRSKNLFMDRMPLFVWSVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPVGGGDPILYQHLF
G6YZTIE02EZ79B      !LSSNIAHAGPSVDLAIFSLHLAGASSILGSINFITTVINMRSEGMRLERIPLFVWAVKITTVLLLLSLPVLAGAITMLLTDRNLNTAFFDPSGGGDPILYQHLF
G6YZTIE02EDO2L      -LSSNIAHAGPSVDLAIFSLHLAGASSILGSINFITTVANMRNEGFRLERIPLFVWAVVITTVLLLLSLPVLAGAITMLLTDRNINTTFFDPAGGGDPVLFQHLF
G6YZTIE02C1VQV      !LAAAIAHAGASVDLGIFSLHLAGVSSILGAVNFMTTVINMQSFGMKMDQMPLFVWSVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPVLYQHLF
G6YZTIE02ELAUF      !LASGVAHAGASVDMGIFSLHLAGVSSILGAINFMTTVINMRSKNLFMDRMPLFVWSVFITAILLLLSLPVLAGAITMLLTDRNLNTSFF!---------------
G6YZTIE02EASMM      !LAAAIAHAGASVDMGIFSLHLAGVSSILGAVNFMTTVINMRSFGMTMDQMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGSGDPVLYQHLF
G6YZTIE02C3CL5      !LAAAIAHAGASVDMGIFSLHLAGVSSILGAVDFMTTVINMRSFGMTMDQMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFD!--------------
G6YZTIE01B6X4W      !LSAGIAHSGPSVDFLIFSLHLLGVGSILGSINYMTTIINMRATGMSFDRMPLFVWAILITSVLLMLSLPVLAGAITMLLTDRNFNTSFFDPAGGGDPVLFQHLF
G6YZTIE01BFE6G      !LAAPIAYAGASVDMGIFSLHLAGVSSILGAVNFMTTVINMRSFGMSMDQMPLFVWAVFITAILLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPILYQHLF
G6YZTIE01BKDOT      !LSAGIAHAGATVDLGISSLHLAGVSSILGAVNFMTTVINMRTTGMTMDRMPLFVWAVFLTAILLLLSLPVLAGAITMLLTDRNLN!------------------
G6YZTIE01BSLKJ      -LSAGIAHSGPLVDFLIFSLHLLGVGSILGSINYMTTIINMRATGMSFDRMSLFVWAILITSVLLMLSLPVLAGAITMLLTGRNFNTSFFDPAGSGDPVLFQHPF
OTU0105_BMOO_07389_T -LSGNLAHAGGSVDLAIFSLHLAGVSSILGAVNFITTIINMRWRGMQFERLPLFVWSVKITAVLLLLSLPVLAGAITMLLTDRNFNTAFFDPAGGG---------
OTU0390_BMOO_06785_G -LAGNLAHAGGSVDLAIFSLHLAGVSSILGAVNFITTIINMRWRGMQFERLPLFVWSVKITAVLLLLSLPVLAGAITMLLTDRNFNTAFFD--------------
OTU0416_BMOO_01113_C -LAGNLAHAGGSVDLAIFSLHLAGVSSILGAVNFITTIINMRWRGMQFERLPLFVWSVKITAILLLLSLPVLAGAITMLLTDRNFNTAFFDPAGGGDPILYQHLF
OTU0270_BMOO_00277_P -LAGNLAHAGGSVDLAIFSLHLAGVSSILGAVNFITTIINMRWRGMQFERLPLFVWSVKITAILLLLSLPVLAGAITMLLTDRNFNTAFFDPAGGGDPILYQHLF
OTU0154_BMOO_07372_C -LSGNLAHAGGSVDLAIFSLHLAGVSSILGAVNFITTILNMRWQGMQFERLPLFVWSVKITAILLLLSLPVLAGAITMLLTDRNFNTSFFDPAGGGDPILYQHL-
OTU0034_BMOO_14161_P -LSSNVAHAGPSVDFAIFSLHLAGVSSILASINFITTAMNMRSSGLRLERVPLFVWSVGITALLLLLSLPVLAGAITMLLTDRNLNTSFFDPAGGGDPILYHHLF
OTU0816_BMOO_06579_T -LAGNLAHAGGSVDLAIFSLHLAGVSSILGAVNFITTIINMRWQGMQFERLPLFVWSVKITAVLLLLSLPVLAGAITMLLTDRNFNT------------------
OTU0926_BMOO_01945_M -LSSNIAHAGASVDLAIFSLHLAGVSSILGAINFISTIFNMRSEGMRLDRMTLFVWSVKITAILLLLSLPVLAGAITMLLTDRNFNTAFFDPAGGGDPILYQHLF
```

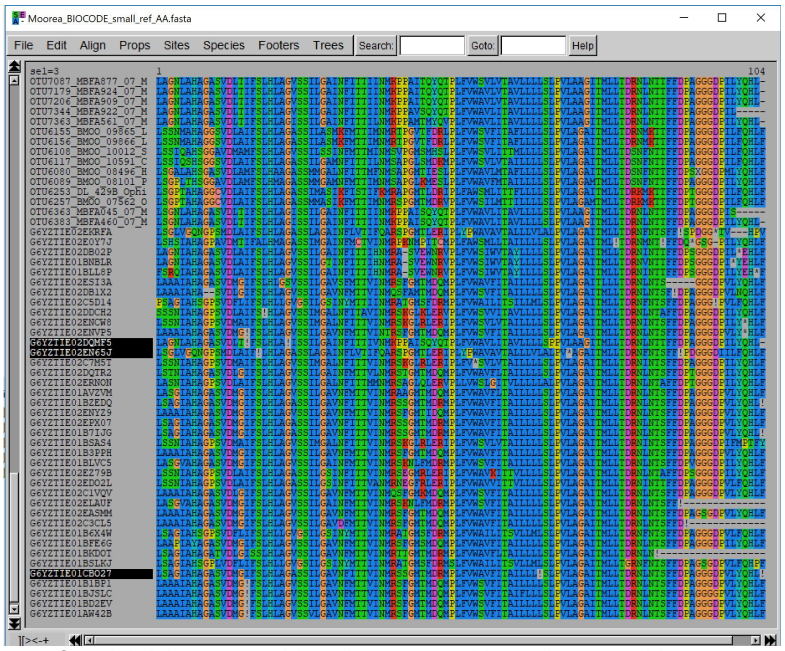
An example of the output of enrichAlignment with rather strict insertion conditions.

Note that in the above example some of the added sequences induce the insertion of a gap in the original alignment (a gapped codon precedes every *OTU* sequences). So, if we would have used the fixed alignment option, most of the sequences would have been discarded due to their frameshift on the first codon position. The **max_NT_trimmed** option is precisely designed to better handle such cases.

On the same dataset, using a fixed alignment (to allow parallelization) and relaxed insertion conditions

- *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 2 -maxINS_inSeq 0 -maxSTOP_inSeq 1 -maxDEL_inSeq 5*

leads to the following result (note that a larger number of sequences were added):



```
sel=3                     1                                                                                  104
OTU7087_MBFA877_07_M      [illegible]
OTU7179_MBFA924_07_M      [illegible]
OTU7206_MBFA909_07_M      [illegible]
OTU7344_MBFA922_07_M      [illegible]
OTU7363_MBFA561_07_M      [illegible]
OTU6155_BMOO_09865_L      [illegible]
OTU6156_BMOO_09866_L      [illegible]
OTU6108_BMOO_10012_S      [illegible]
OTU6117_BMOO_10591_C      [illegible]
OTU6080_BMOO_08496_H      [illegible]
OTU6089_BMOO_08101_P      [illegible]
OTU6253_DL_429B_Ophi      [illegible]
OTU6257_BMOO_07562_O      [illegible]
OTU6363_MBFA045_07_M      [illegible]
OTU6383_MBFA460_07_M      [illegible]
G6YZTIE02EKRFA            [illegible]
G6YZTIE02E0Y7J            [illegible]
G6YZTIE02DB02P            [illegible]
G6YZTIE01BNBLR            [illegible]
G6YZTIE01BLL8P            [illegible]
G6YZTIE02ESI3A            [illegible]
G6YZTIE02DB1X2            [illegible]
G6YZTIE02C5D14            [illegible]
G6YZTIE02DDCH2            [illegible]
G6YZTIE02ENCW8            [illegible]
G6YZTIE02ENVP5            [illegible]
G6YZTIE02DQMF5            [illegible]
G6YZTIE02EN65J            [illegible]
G6YZTIE02C7M5T            [illegible]
G6YZTIE02DQTR2            [illegible]
G6YZTIE02ERNON            [illegible]
G6YZTIE01AVZVM            [illegible]
G6YZTIE01BZEDQ            [illegible]
G6YZTIE02ENYZ9            [illegible]
G6YZTIE02EPX07            [illegible]
G6YZTIE01B7IJG            [illegible]
G6YZTIE01BSAS4            [illegible]
G6YZTIE01B3PPH            [illegible]
G6YZTIE01BLVC5            [illegible]
G6YZTIE02EZ79B            [illegible]
G6YZTIE02EDO2L            [illegible]
G6YZTIE02C1VQV            [illegible]
G6YZTIE02ELAUF            [illegible]
G6YZTIE02EASMM            [illegible]
G6YZTIE02C3CL5            [illegible]
G6YZTIE01B6X4W            [illegible]
G6YZTIE01BFE6G            [illegible]
G6YZTIE01BKDOT            [illegible]
G6YZTIE01BSLKJ            [illegible]
G6YZTIE01CBO27            [illegible]
G6YZTIE01B1BP1            [illegible]
G6YZTIE01BJSLC            [illegible]
G6YZTIE01BD2EV            [illegible]
G6YZTIE01AW42B            [illegible]
```

An example of the output of enrichAlignment with rather relaxed insertion conditions.

Using these options more sequences were added but some of them contain multiple internal frameshifts and stop codons. Note that even those sequences are correctly aligned by MACSE.

5. Understanding why some sequences are not added and traceability options

The **enrichAlignment** MACSE subprogram not only produces the two usual FASTA output files containing respectively the nucleotide and amino acid alignments, but also a tabular text file.

- *java -jar macse.jar -prog enrichAlignment -align align.fasta -seq sequences.fasta -out_tested_seq_info output_stats.csv*

This file contains, for each tested sequence (one per line), the following fields:

- *Sequence name*
- *Whether the sequence was added or not*
- *Number of frameshifts, trimmed (when sequences are alterable) or not*
- *Number of trimmed frameshifts*
- *Number of stop codons*
- *Number of amino acid deletions (in the sequence)*
- *Number of amino acid insertions (internal only)*
- *Number of amino acid insertions (internal or at the alignment extremities)*

If you wonder why a sequence was not added to an alignment, you can force its insertion by using relaxed parameters that will allow multiple frameshifts, stops and insertion/deletion events and visualize its alignment.

6. Other options

You can find other options related to this program from the following links :

- genetic codes
- alignment costs
- compressed AA alphabet
- allowed nucleotides
- refineAlignment: MACSE subprogram to improve an existing alignment

Alignment export

MACSE pinpoints frameshifts using the "!" character. However this is not standard usage and alignment with such characters will be rejected by most software that take a multiple sequence alignment as input. This MACSE subprogram allows to replace "!" characters in nucleotide and amino acid alignments. It also allows computing some basic statistics about the number insertions, deletions, frameshifts, and stop codons per sequence.

This export procedure as several advantages over a simple "search and replace" of the "!" character in your FASTA files:

- 1. you can choose to modify the whole codons containing frameshifts (e.g. replacing AT! by either --- or NNN) rather than just acting on a single character (e.g. replacing AT! by either AT- or ATN) - 2. you can handle differently frameshifts that occur within sequences and those at sequence extremities - 3. you can choose to transform internal stop codons into NNN codons - 4. if after replacement you have useless sites entirely made of gap-only codons (---) those sites will be removed.

URL : samples/exportAlignment/

1. Replacing frameshifts and stop codons

As a stop codon is expected at the end of full coding sequences, you can handle differently the stop codons appearing at the end of the sequences and those appearing within the sequences.

- *java -jar macse.jar -prog exportAlignment -align align.fasta -codonForFinalStop ---*
- *java -jar macse.jar -prog exportAlignment -align align.fasta -codonForInternalStop NNN*

Similarly, terminal frameshifts (those at the end of the sequences) often reflect the fact that the coding sequence is incomplete rather than the fact that it contains an error or a frameshift. You can hence handle them differently.

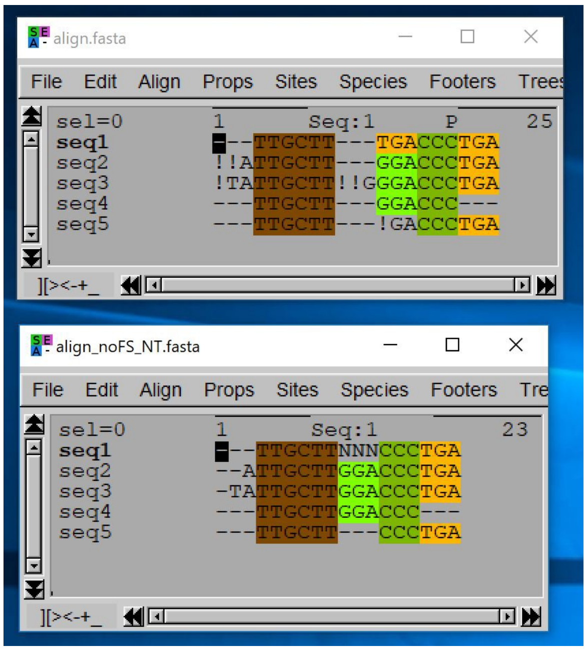
- *java -jar macse.jar -prog exportAlignment -align align.fasta -codonForExternalFS ---*
- *java -jar macse.jar -prog exportAlignment -align align.fasta -codonForInternalFS NNN*

Finally, you can specify how to handle frameshift characters that still remain after replacement performed at the codon level.

- *java -jar macse.jar -prog exportAlignment -align align.fasta -charForRemainingFS -*

Export guidelines: a reasonable export solution is to preserve final stop codons (as they are expected), to replace other stop codon by "NNN", to replace unexpected internal frameshift codons by "---" (or "NNN") and to replace remaining frameshift characters (those at sequence extremities) by "-".

- *java -jar macse.jar -prog exportAlignment -align align.fasta -codonForInternalStop NNN -codonForInternalFS --- -charForRemainingFS - -out_NT align_noFS_NT.fasta -out_AA align_noFS_AA.fasta*


align.fasta
File Edit Align Props Sites Species Footers Trees
align_noFS_NT.fasta
File Edit Align Props Sites Species Footers Tre

```
sel=0      1      Seq:1      P    25
seq1       ---TTGCTT---TGACCCTGA
seq2       !!ATTGCTT---GGACCCTGA
seq3       !TATTGCTT!!GGGACCCTGA
seq4       ---TTGCTT---GGACCC---
seq5       ---TTGCTT---!GACCCTGA
```

```
sel=0      1      Seq:1       23
seq1       ---TTGCTTNNNCCCTGA
seq2       --ATTGCTTGGACCCTGA
seq3       -TATTGCTTGGACCCTGA
seq4       ---TTGCTTGGACCC---
seq5       ---TTGCTT---CCCTGA
```

Example of export procedure using export guidelines, TGA is a stop codon.

2. Getting some statistics on your (exported) alignment

The following option creates a tabular CSV file containing the number of A, C, G, T, -, and ! characters per site. This could be useful to estimate, for instance, GC or GC3 contents.

- *java -jar macse.jar -prog exportAlignment -align align.fasta -out_stat_per_site output_frequencies.csv*

You can also obtain a tabular CSV file containing the number of internal frameshifts, stop codons, and deletions for each sequence (the counts are done at the codon/AA level).

- *java -jar macse.jar -prog exportAlignment -align align.fasta -out_stat_per_seq output_stats.csv*

Those options can be used simultaneously with other export options (e.g. *codonForInternalFS*). In such a case, the statistics are computed after applying the requested replacements.

3. Having a unique codon for each amino acid

You can use *exportAlignment* to *canonize* your alignment. In this case, your alignment will be transformed so that each amino acid will be consistently encoded with the same codon. This could be useful to generate a consensus nucleotide sequence reflecting amino acid frequencies if the consensus solution provided by *exportAlignment* does not exactly fit your need.

- *java -jar macse.jar -prog exportAlignment -align align.fasta -canonize_ON*

4. Computing a consensus sequence

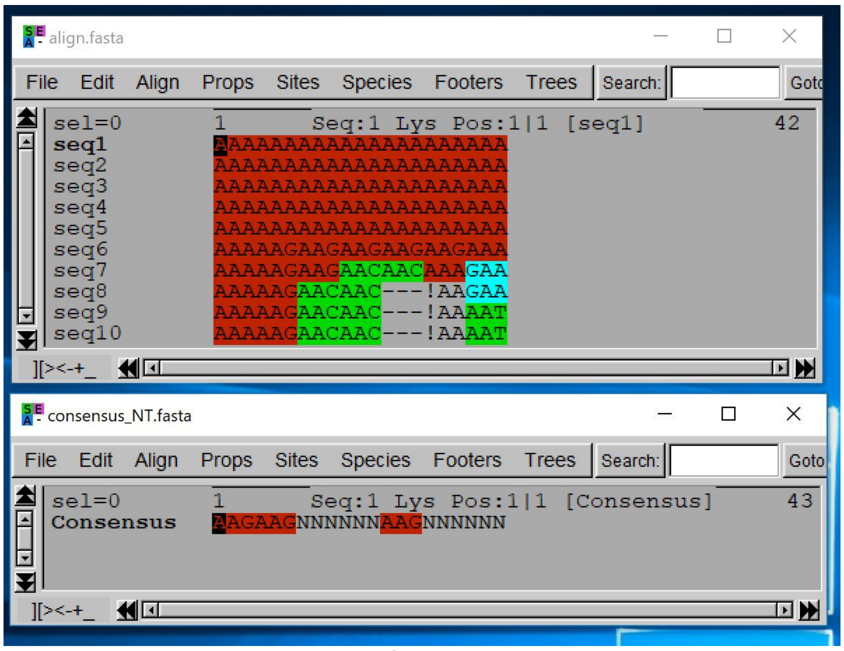
A consensus sequence is a way to summarize an alignment into a single sequence. At the amino acid level, such a sequence can be obtained by keeping for each site the most frequent amino acid. However, if even the most frequent amino acid is not that frequent, you may prefer to have an unknown amino acid 'X' at this position. MACSE lets you choose the minimum frequency (percentage) that an amino acid should reach to be kept in the consensus sequence. It also provides a nucleotide consensus sequence such that, if you translate it using the default genetic code, you get the consensus amino acid sequence of your alignment.

- *java -jar macse.jar -prog exportAlignment -align align.fasta -cons_threshold 0.71 -out_NT_consensus consensus_NT.fasta -out_AA_consensus consensus_AA.fasta -name_cons_seq Consensus*

The **name_cons_seq** option allows to specify the name of the consensus.

The figure below shows the input alignment provided to the above command and the resulting consensus sequence. Note that with a threshold of 0.71 and 10 sequences, an amino acid has to appear in at least 8 sequences to be present in the consensus sequence. The Aspargine can be encoded by AAA or AAG. The first and second amino acids are thus preserved in the consensus sequence, since in both cases the same amino acid is present in 100% of the sequences (though encoded using two different codons in the second case). The codon used in the consensus nucleotide sequence is the codon used by MACSE to encode Aspargine every time it appears in a consensus sequence (not necessarily the most frequent codon at this position of the input alignment). The fifth amino acid is also preserved as Aspargine is present 6 times out of 7 at this position (gaps are ignored in the frequency computation) and 6/7>0.71.

Finally, for the last three positions of the alignment, the most frequent nucleotides (AAT) each appear in more than 80% of the sequences. However, no amino acid appears in more that 70% of the sequences. This illustrates the difference between a consensus sequence built at the nucleotide level that will return AAT and the consensus at the amino acid level done here by MACSE that returns NNN.


align.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto

```
sel=0        1      Seq:1 Lys Pos:1|1 [seq1]      42
seq1         AAAAAAAAAAAAAAAAAAAAA
seq2         AAAAAAAAAAAAAAAAAAAAA
seq3         AAAAAAAAAAAAAAAAAAAAA
seq4         AAAAAAAAAAAAAAAAAAAAA
seq5         AAAAAAAAAAAAAAAAAAAAA
seq6         AAAAAGAAGAAGAAGAAGAAA
seq7         AAAAAGAAGAACAACAAAGAA
seq8         AAAAAGAACAAC---!AAGAA
seq9         AAAAAGAACAAC---!AAAAT
seq10        AAAAAGAACAAC---!AAAAT
```

consensus_NT.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto

```
sel=0        1      Seq:1 Lys Pos:1|1 [Consensus]      43
Consensus    AAGAAGNNNNNNAAGNNNNNN
```

Example of a consensus sequence with a threshold of 0.71

5. Related documentation

You can find other options related to this program from the following links:

- genetic codes
- compressed AA alphabet
- allowed nucleotides

Run multiple programs

URL : samples/multiPrograms/

This subprogram allows to sequentially executes multiple MACSE commands contained in a text file (one per line). This allows basic scripting for non bioinformaticians. The main option of this subprogram is a file containing a list of MACSE commands. Each line of this file must contains 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 this command file.

You can use the GUI to easily generate the command line you need on a single example, copy this command (using copy/paste or the copy to clipboard button) and adapt this command to run it on multiple datasets using this subprogram or the command line facilities of MACSE.

You can also chain multiple analysis on a single datafile using this option and use the parameter file to ensure reproductibility and traceability.

Here is a simple example of the program usage!

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

Refining alignments

The **refineAlignment** subprogram tries to further improve an existing nucleotide alignment. It aligns sequences at the nucleotide level while scoring the considered nucleotide alignments based on their amino acid translation. It thus favors nucleotide gap stretches that are multiple of three but it also considers those inducing frameshifts, when they allow to recover the underlying codon structure. It therefore produces alignments which benefit from the higher similarity of amino acid sequences while accounting for frameshifts and stop codons that could occur in pseudogenes or in poor quality sequences (alignment costs).

This program is highly related to **alignSequences**. The main difference is that alignSequences takes unaligned sequences as input and thus need to build a first alignment before improving it, while refineAlignment starts from an alignment as input. Those two programs share most of their options and it is strongly advised to read the alignSequences help pages before using **refineAlignment**.

refineAlignment can be used to improve an alignment produced by MACSE (e.g. produced by alignSequences with prameters favoring speed over accuracy) or by any other alignment software.

- On one hand, if you used a 3-step approach to align your sequences (1/ translating all coding NT into AA, 2/ aligning those AA sequences, and 3/ using the obtained amino acid alignment to derive the NT one) then efineAlignment can be relevant to rapidly identify sequences with frameshits that have been erroneously aligned using this strategy. We however strongly suggest to use alignSequences with adequate speed options to spot frameshifts prior to using this 3 steps strategy.
- On the other hand, if you have simply aligned your sequences at the nucleotide level, the resulting alignment can be so bad (at least in regard to the underlying codon structure) that refineAlignment will be very slow and may even worsen the input alignment.

Folder: samples/refineAlignment/

1. Some basic usage examples

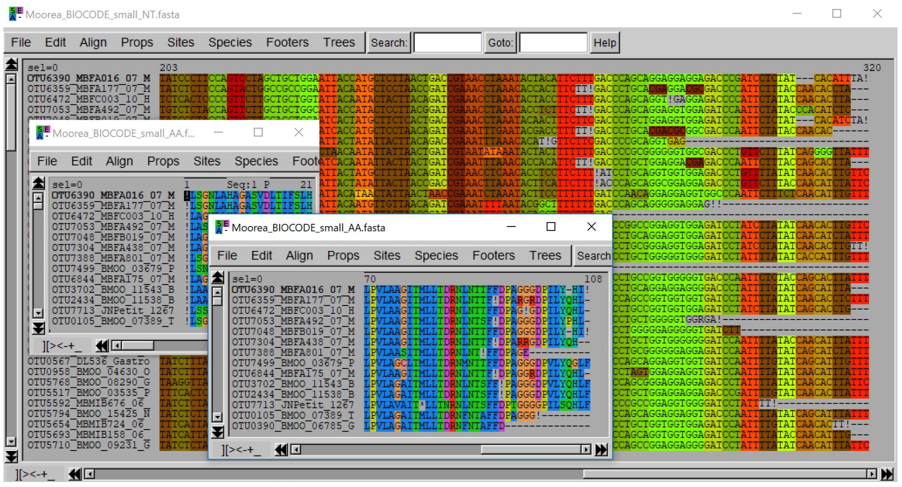
The same options as those used for alignSequences allow you to (1) control for the balance between optimization and speed (**-optim**, **-max_refine_iter**, **local_realign_init**, **local_realign_dec**), (2) specify a subset of less reliable sequences with different costs for their frameshifts and stop codons (**-seq_lr**), (3) choose the name of output files (**out_NT**, **out_AA**), and (4) select your own elementary alignment costs:

- *java -jar macse.jar -prog refineAlignment -align align.fasta -optim 1*
- *java -jar macse.jar -prog refineAlignment -align align.fasta -optim 2*
- *java -jar macse.jar -prog refineAlignment -optim 2 -align align.fasta -out_NT output_NT.fasta -out_AA output_AA.fasta*
- *java -jar macse.jar -prog refineAlignment -align align.fasta -optim 2 -seq sequences.fasta -seq_lr sequences_lr.fasta*

2. A more realistic example related to metabarcoding

Metabarcoding analysis often requires to handle thousands of sequences. Such datasets are not directly tractable with the alignSequence subprogram of MACSE, but they can be handled by sequentially adding your newly obtained sequences to a reference alignment containing protein-coding sequences of related taxa for your targeted locus (COX1, matK, rbcL, etc...). We successfully used this approach in the Moorea project, Leray et al. 2013(https://frontiersinzoology.biomedcentral.com/articles/10.1186/1742-9994-10-34). In this project, we initially got an external alignment (from previous studies) of about 7,000 COX1 sequences. In a fisrt step, we used refineAlignment to identify potential errors in this alignment. About 200 sequences were detected as having unexpected frameshifts or stop codons. In the following example, the analysis is shown on a small subset of the original alignment:

- *java -jar macse.jar -prog refineAlignment -align Moorea_BIOCODE_small.fasta -gc_def 5 -optim 1 -max_refine_iter 1*


Moorea_BIOCODE_small_NT.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto: Help
Moorea_BIOCODE_small_AA.f...
Moorea_BIOCODE_small_AA.fasta

Example of refineAlignment output on a COX1 alignment.

In this example, the input alignment mostly respects the codon structure but it started on the third reading frame. **refineAlignment** automatically detects this and adds the required frameshits at the beginning of the sequences. Moreover, it improves the alignment of a dozen of sequences (appearing together in the fasta file and hence easier to spot) by introducing some frameshifts near the sequence ends. Those sequences, with few frameshifts and stop codons, can then be kept or (manually) removed depending on your objective.

The frameshifts at the extremities of the sequences can be replaced by gap codons using exportAlignment. This is probably better if all first or all last codons are frameshifts since such configurations can distorted the count of frameshift events done by enrichAlignment when conditionally adding new sequences.

- *java -jar macse.jar -prog exportAlignment -align Moorea_BIOCODE_small_ref.fasta -codonForExternalFS --- -codonForInternalFS ---*

3. Related documentation

You can find other options related to this program from the following links:

- genetic codes
- alignment costs
- compressed AA alphabet
- allowed nucleotides
- MACSE subprogram to align sequences(./alignSequences.html)

Deriving a nucleotide alignment from an amino acid alignment

The alignSequences subprogram could be time consuming for large datasets. A possible, still efficient, strategy for large datasets is the following:

- 1. use trimNonHomologousFragments to eliminate non-homologous sequence fragments
- 2. use alignSequences with rapid optimization options to unravel frameshifts
- 3. preserve frameshifts (but not gaps) to obtained unaligned nucleotide sequences with documented frameshifts (simply delete the '-' from your FASTA file)
- 4. translate those nucleotide sequences into amino acid sequences using translateNT2AA.
- 5. align these amino acids sequences with your favorite alignment software (e.g. MUSCLE, PRANK, MAFFT)
- 6. use reportGapsAA2NT to derive your nucleotide alignment from the amino acid alignment found at step 5.

This pipeline as been successfully used to produce OrthoMaM alignments and is available through a dedicated web service at http://mbb.univ-montp2.fr/MBB/.

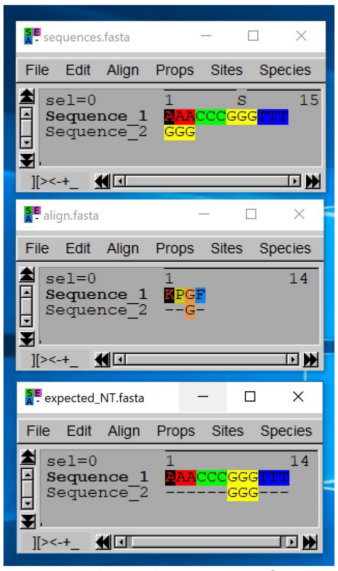
1. Reporting gaps from aligned amino acid sequences to unaligned nucleotide sequences

The reportGapsAA2NT takes as input a FASTA file with unaligned nucleotide sequences and a FASTA file of aligned sequences that are the amino acid translations of the nucleotide ones. Each sequence should hence be present with the exact same name in both files and should be three time longer in the nucleotide file than in the amino acid file (ignoring gaps).

Warning: For the sequence lengths to match, you should either remove any final stop codons from your nucleotide sequences or translate them into the unkwown amino acid 'X'.

URL : samples/reportGapsAA2NT/

- *java -jar macse.jar -prog reportGapsAA2NT -align_AA align.fasta -seq sequences.fasta*
- *java -jar macse.jar -prog reportGapsAA2NT -align_AA align.fasta -seq sequences.fasta -out_NT output_NT.fasta*


sequences.fasta
File Edit Align Props Sites Species
align.fasta
File Edit Align Props Sites Species
expected_NT.fasta
File Edit Align Props Sites Species

```
sel=0          1      S      15
Sequence_1     AAACCCGGGTTT
Sequence_2     GGG
```

```
sel=0          1            14
Sequence_1     KPGF
Sequence_2     --G-
```

```
sel=0          1            14
Sequence_1     AAACCCGGGTTT
Sequence_2     ------GGG---
```

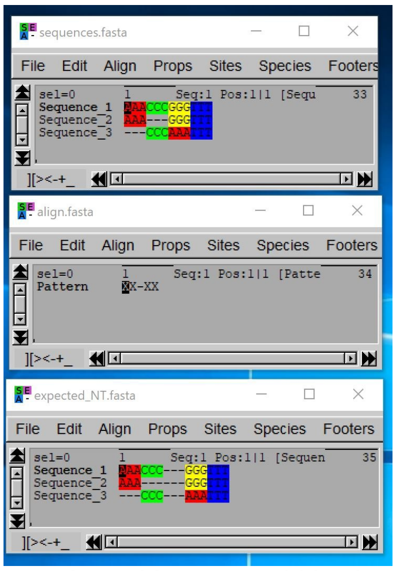
Given unaligned nucleotide sequences -top- and the alignment of their amino acid translations with the exact same names -middle- you can derive a nucleotide sequence alignment -bottom-

2. Reporting gaps from a sequence pattern to a nucleotide alignment

This option could be useful to build an alignment with a very large number of sequences: you can start by clustering them (either based on homology or taxonomy), then aligning independently each cluster of sequences and produce a codon consensus sequence for each aligned cluster (exportAlignment, align those consensus sequences, and use the resulting alignment to produce the global alignment by reporting gaps of each consensus sequence into the corresponding cluster alignment. In such case, the aligned consensus sequence is thus seen as a pattern indicating where gaps should be added to the cluster alignment in order to build the global alignment.

- *java -jar macse.jar -prog reportGapsAA2NT -align_AA align.fasta -seq sequences.fasta -AA_seq_as_pattern_ON*

Warning: with this option, the **sequences.fasta** file must contain aligned nucleotide sequences and the **align.fasta** file must only contain one sequence that serves as pattern to indicate where gaps should be inserted in the input nucleotide alignment:


sequences.fasta
File Edit Align Props Sites Species Footers
align.fasta
File Edit Align Props Sites Species Footers
expected_NT.fasta
File Edit Align Props Sites Species Footers

```
sel=0          1         Seq:1 Pos:1|1 [Sequ      33
Sequence_1     AAACCCGGGTTT
Sequence_2     AAA---GGGTTT
Sequence_3     ---CCCAAATTT
```

```
sel=0          1         Seq:1 Pos:1|1 [Patte     34
Pattern        XX-XX
```

```
sel=0          1         Seq:1 Pos:1|1 [Sequen    35
Sequence_1     AAACCC---GGGTTT
Sequence_2     AAA------GGGTTT
Sequence_3     ---CCC---AAATTT
```

An example of the gap reporting using an amino acid pattern -middle- instead of an amino acid alignment.

3. Related documentation

You can find other options related to this program from the following link:

- nucleotides

Report amino acids mask on nucleotides sequences

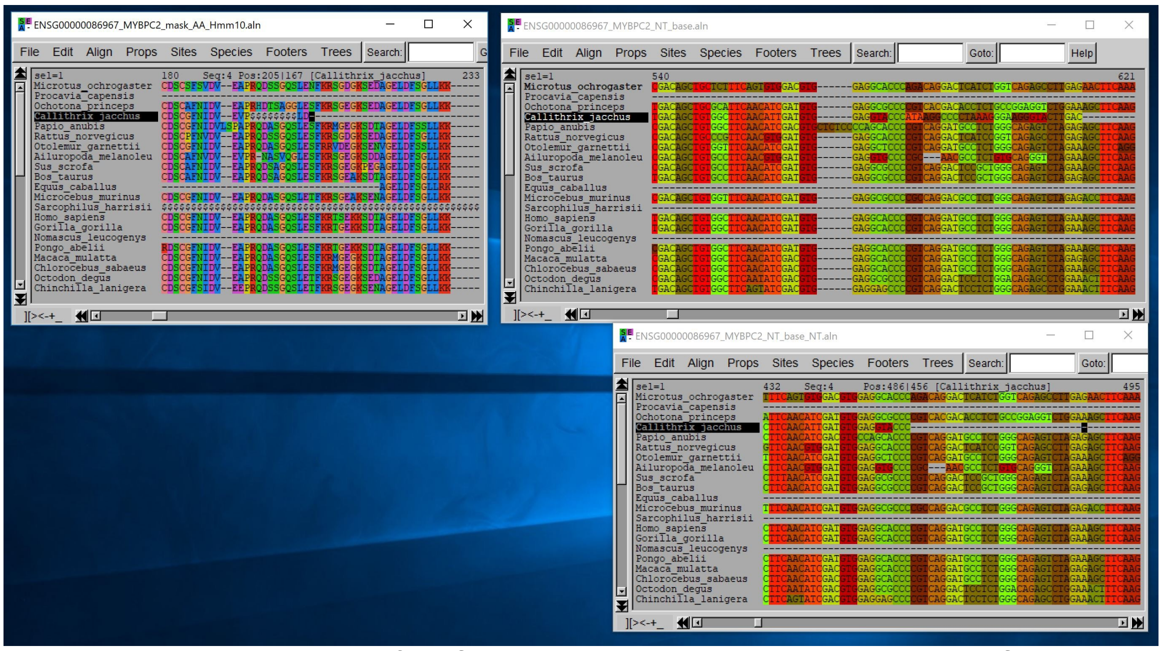
If your analyses are sensitive to alignment errors (e.g. dN/dS estimation), we strongly advice to use a post filtering of your alignment at the amino acid level (e.g. using HMMCleaner, BMGE or trimAl) and to report this AA masking/filtering at the nucleotide level using .

This subprogram is dedicated to this task. It uses a nucleotide alignment and a filtered (masked) version of its amino acid translation to derived the filtered version of the input nucleotide alignment.

Warning By default some post processings are done to also mask isolated codons (those surrounded only by gaps or masked codons) and sequences with not enough remaining codons can also be completely removed from the alignment.

Here is an example with some of the key options of this program. The input file is a fasta file with the aligned nucleotide sequences and the masked version of the corresponding amino acid alignment. The masking has been done with **HMMCleaner**, the $ symbol indicate the amino acids masked by HMMCleaner:

- *java -jar macse.jar -prog reportMaskAA2NT -align_AA ENSG00000086967_MYBPC2_mask_AA_Hmm10.aln -align ENSG00000086967_MYBPC2_NT_base.aln -min_NT_to_keep_seq 30 -mask_AA $ -min_seq_to_keep_site 4 -min_percent_NT_at_ends 0.3 -dist_isolate_AA 3 -min_homology_to_keep_seq 0.3 -min_internal_homology_to_keep_seq 0.5 -out_stat_file stat.csv*


ENSG00000086967_MYBPC2_mask_AA_Hmm10.aln
File Edit Align Props Sites Species Footers Trees Search:
ENSG00000086967_MYBPC2_NT_base.aln
File Edit Align Props Sites Species Footers Trees Search: Goto: Help
ENSG00000086967_MYBPC2_NT_base_NT.aln
File Edit Align Props Sites Species Footers Trees Search: Goto:

```
sel=1                 180    Seq:4 Pos:205|167 [Callithrix_jacchus]      233
Microtus_ochrogaster  CDSCSFSVDV--EAPRQDSSGQSLENFKRSGDGKSEDAGELDFSGLLKK-----
Procavia_capensis     ------------------------------------------------------
Ochotona_princeps     CDSCAFNIDV--EAPRHDTSAGGLESFKRSGEGKSEDAGELDFSGLLKK-----
Callithrix_jacchus    CDSCGFNIDV--EVP$$$$$$$$LD-----------------------------
Papio_anubis          CDSCGFNIDVLSPAPRQDASGQSLESFKRMGEGKSDTAGELDFSSLLKK-----
Rattus_norvegicus     CDSCPFNVDV--EAPRQDSSGQSLESFKRSGDGKSEDAGELDFSGLLKK-----
Otolemur_garnettii    CDSCGFNIDV--EAPRQDASGQSLESFRRVDEGKSENVGELDFSSLLKK-----
Ailuropoda_melanoleu  CDSCAFNVDV--EVPR-NASVQGLESFKRSGEGKSDDAGELDFSGLLKK-----
Sus_scrofa            CDSCAFNIDV--EAPRQDSAGQSLESFKRSGEGKPEGAGELDFSGLLKK-----
Bos_taurus            CDSCAFNIDV--EAPRQDSAGQSLESFKRSGEAKSDTAGELDFSGLLKK-----
Equus_caballus        -------------------------------------AGELDFSGLLRK-----
Microcebus_murinus    CDSCGFNIDV--EAPRQDASGQSLETFKRSGEAKSENAGELDFSGLLKK-----
Sarcophilus_harrisii  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Homo_sapiens          CDSCGFNIDV--EAPRQDASGQSLESFKRTSEKKSDTAGELDFSGLLKK-----
Gorilla_gorilla       CDSCGFNIDV--EAPRQDASGQSLESFKRTGEKKSDTAGELDFSGLLKK-----
Nomascus_leucogenys   ------------------------------------------------------
Pongo_abelii          RDSCGFNIDV--EAPRQDASGQSLESFKRTGEKKSDTAGELDFSGLLKK-----
Macaca_mulatta        CDSCGFNIDV--EAPRQDASGQSLESFKRMGEGKSDTAGELDFSGLLKK-----
Chlorocebus_sabaeus   CDSCGFNIDV--EAPRQDASGQSLESFKRMGEGKSDTAGELDFSGLLKK-----
Octodon_degus         CDSCGFNIDV--EAPRQDSSGQSLETFKRSGEGKSEDAGELDFSGLLKK-----
Chinchilla_lanigera   CDSCGFSIDV--EEPRQDSSGQSLETFKRSGEGKSENAGELDFSGLLKK-----
```

```
sel=1                 540                                                                                621
Microtus_ochrogaster  CGACAGCTGCTCTTTCAGTGTGGACGTG------GAGGCACCCAGACAGGACTCATCTGGTCAGAGCCTTGAGAACTTCAAA
Procavia_capensis     ----------------------------------------------------------------------------------
Ochotona_princeps     TGACAGCTGCGCATTCAACATCGATGTG------GAGGCGCCCCGTCACGACACCTCTGCCGGAGGTCTGGAAAGCTTCAAG
Callithrix_jacchus    TGACAGCTGTGGCTTCAACATTGATGTG------GAGGTACCCATAAGGCCCCTAAAGGGAAGGGTACTTGAC---------
Papio_anubis          CGACAGCTGTGGCTTCAACATCGACGTGCTCTCCCCAGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAGAGCTTCAAG
Rattus_norvegicus     CGACAGCTGCCGTTCAACGTGGATGTG------GAGGCACCCCGTCAGGACTCATCCGGTCAGAGCCTTGAGAGCTTCAAG
Otolemur_garnettii    CGACAGCTGTGGTTTCAACATCGATGTG------GAGGCTCCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAGG
Ailuropoda_melanoleu  CGACAGCTGTGCCTTCAACGTGGATGTG------GAGGTGCCCCGC---AACGCCTCTGTGCAGGGTCTAGAAAGCTTCAAG
Sus_scrofa            CGACAGCTGTGCCTTTAACATCGATGTG------GAGGCGCCCCGTCAGGACTCCGCTGGGCAGAGTCTAGAAAGCTTCAAG
Bos_taurus            TGACAGCTGTGCCTTCAACATCGATGTG------GAGGCGCCCCGTCAGGACTCCGCTGGGCAGAGTCTAGAGAGCTTCAAG
Equus_caballus        ----------------------------------------------------------------------------------
Microcebus_murinus    CGACAGCTGTGGTTTCAACATCGATGTG------GAGGCGCCCCGCCAGGACGCCTCTGGGCAGAGTCTAGAGACCTTCAAG
Sarcophilus_harrisii  ----------------------------------------------------------------------------------
Homo_sapiens          TGACAGCTGTGGCTTCAACATCGATGTG------GAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Gorilla_gorilla       TGACAGCTGTGGCTTCAACATCGATGTG------GAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Nomascus_leucogenys   ----------------------------------------------------------------------------------
Pongo_abelii          CGACAGCTGTGGCTTCAACATCGATGTG------GAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Macaca_mulatta        CGACAGCTGTGGCTTCAACATCGACGTG------GAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAGAGCTTCAAG
Chlorocebus_sabaeus   CGACAGCTGTGGCTTCAACATCGACGTG------GAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Octodon_degus         CGACAGCTGTGGCTTCAATATCGACGTG------GAGGCGCCCCGTCAGGACTCCTCTGGACAGAGCCTGGAAACTTTCAAG
Chinchilla_lanigera   TGACAGCTGTGGCTTCAGTATCGACGTG------GAGGAGCCCCGTCAGGACTCCTCTGGGCAGAGCCTGGAAACTTTCAAG
```

```
sel=1                 432    Seq:4      Pos:486|456 [Callithrix_jacchus]         495
Microtus_ochrogaster  TTTCAGTGTGGACGTGGAGGCACCCAGACAGGACTCATCTGGTCAGAGCCTTGAGAACTTCAAA
Procavia_capensis     ----------------------------------------------------------------
Ochotona_princeps     ATTCAACATCGATGTGGAGGCGCCCCGTCACGACACCTCTGCCGGAGGTCTGGAAAGCTTCAAG
Callithrix_jacchus    CTTCAACATTGATGTGGAGGTACCC---------------------------------------
Papio_anubis          CTTCAACATCGACGTGCCAGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAGAGCTTCAAG
Rattus_norvegicus     GTTCAACGTGGATGTGGAGGCACCCCGTCAGGACTCATCCGGTCAGAGCCTTGAGAGCTTCAAG
Otolemur_garnettii    TTTCAACATCGATGTGGAGGCTCCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAGG
Ailuropoda_melanoleu  CTTCAACGTGGATGTGGAGGTGCCCCGC---AACGCCTCTGTGCAGGGTCTAGAAAGCTTCAAG
Sus_scrofa            CTTTAACATCGATGTGGAGGCGCCCCGTCAGGACTCCGCTGGGCAGAGTCTAGAAAGCTTCAAG
Bos_taurus            CTTCAACATCGATGTGGAGGCGCCCCGTCAGGACTCCGCTGGGCAGAGTCTAGAGAGCTTCAAG
Equus_caballus        ----------------------------------------------------------------
Microcebus_murinus    TTTCAACATCGATGTGGAGGCGCCCCGCCAGGACGCCTCTGGGCAGAGTCTAGAGACCTTCAAG
Sarcophilus_harrisii  ----------------------------------------------------------------
Homo_sapiens          CTTCAACATCGATGTGGAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Gorilla_gorilla       CTTCAACATCGATGTGGAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Nomascus_leucogenys   ----------------------------------------------------------------
Pongo_abelii          CTTCAACATCGATGTGGAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Macaca_mulatta        CTTCAACATCGACGTGGAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAGAGCTTCAAG
Chlorocebus_sabaeus   CTTCAACATCGACGTGGAGGCACCCCGTCAGGATGCCTCTGGGCAGAGTCTAGAAAGCTTCAAG
Octodon_degus         CTTCAATATCGACGTGGAGGCGCCCCGTCAGGACTCCTCTGGACAGAGCCTGGAAACTTTCAAG
Chinchilla_lanigera   CTTCAGTATCGACGTGGAGGAGCCCCGTCAGGACTCCTCTGGGCAGAGCCTGGAAACTTTCAAG
```

Note how the isolated LD amino acids of the Callithrix sequence though not masked by HMMCleaner are masked by the default post-processing of MACSE

2. Masking traceability

A FASTA file containing the detail of the masking process is also output. In this FASTA unmasked nucleotide are in capital letters while masked ones are in lower case.

3. Related documentation

You can find other options related to this program from the following links:

- allowed nucleotides
- trimAlignment: trims the input alignment by removing gappy sites at the beginning/end of the alignment
- trimNonHomologousFragments: identifies sequence fragments that do not share homology with other sequences and remove those fragments.

Spliting alignment or extracting a sub-alignment (subset of species and/or sites)

Given a large alignment, one can be interested in only a subset of the aligned sequences or in a subset of the sites (i.e. some specific regions/domains of the CDS).

The MACSE subprogram **splitAlignment** is designed to extract sub-alignments you are really interested in while 1/ removing useless sites, i.e. those only made of gaps after the restriction, and 2/ preserving the codon structure in the resulting sub-alignment since "!" and "-" characters are handled differently.

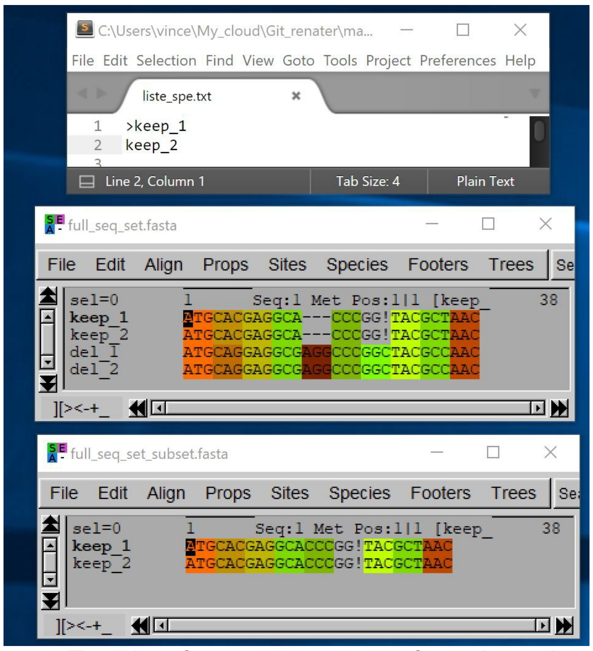
WARNING The codon structure is preserved only if, following MACSE convention, the input alignment contains frameshift characters ("!") and not classical gap characters ("-") to indicate frameshifts in the nucleotide alignment.

Folder: samples/splitAlignment/

1. Extracting an alignment containing a subset of sequences

The **subset** option allows you to specify which species should be kept in the sub-alignment by taking as argument a text file containing their names. Each line of this text file must contain the name of a sequence, preceded or not by the ">" symbol:

- *java -jar macse.jar -prog splitAlignment -align full_seq_set.fasta -subset liste_spe.txt*


C:\Users\vince\My_cloud\Git_renater\ma...
File Edit Selection Find View Goto Tools Project Preferences Help
liste_spe.txt
Line 2, Column 1
Tab Size: 4
Plain Text
full_seq_set.fasta
File Edit Align Props Sites Species Footers Trees Se
full_seq_set_subset.fasta
File Edit Align Props Sites Species Footers Trees Se

```
>keep_1
keep_2
```

```
sel=0        1      Seq:1 Met Pos:1|1 [keep_      38
keep_1       ATGCACGAGGCA---CCCGG!TACGCTAAC
keep_2       ATGCACGAGGCA---CCCGG!TACGCTAAC
del_1        ATGCAGGAGGCGAGGCCCGGCTACGCCAAC
del_2        ATGCAGGAGGCGAGGCCCGGCTACGCCAAC
```

```
sel=0        1      Seq:1 Met Pos:1|1 [keep_      38
keep_1       ATGCACGAGGCACCCGG!TACGCTAAC
keep_2       ATGCACGAGGCACCCGG!TACGCTAAC
```

A simple example of splitAlignment. Top: list of sequence names of species to keep in the alignment subset; Middle: input alignment; Bottom: resulting sub-alignment

This could be especially useful when you download alignments from public databases such as EnsEMBL or OrthoMaM for which you don't have a precise control over the species included. For instance if you download an alignment of 1:1 orthologous mammalian sequences from OrthoMaM, you can easily restrict this alignment to Primates by providing the primate species names in a separate file (in OrthoMaM alignments, sequences are named according to the species they come from):

- *java -jar macse.jar -prog splitAlignment -align TMEM184_Ensembl_Plos.fasta -subset primates.txt*

Note that by default the sub-alignment containing the sequences to kept is saved in a FASTA file whose name end with "_subset.fasta". The sub-alignment containing all other sequences is also generated and is stored in a FASTA file with a name ending by "_others.fasta". Of course you can specify different output file names:

- *java -jar macse.jar -prog splitAlignment -align align.fasta -subset species.txt -out_subset align_subset.fasta -out_others align_others.fasta*

2. Extracting an alignment containing a subset of sites

You can also restrict your alignment to a subset of sites by indicating the region to keep. Here is a basic example :

- *java -jar macse.jar -prog splitAlignment -align align.fasta -first_site 4 -last_site 9*

If you provide only the first (resp. last) site to keep, the **last_site** (resp. **first_site**) will stay fixed at the default value, wich is the end (resp. beginning) of the alignment.

In case you want to preserve several regions of your alignment, you can use a file containing a list of the site intervals to keep:

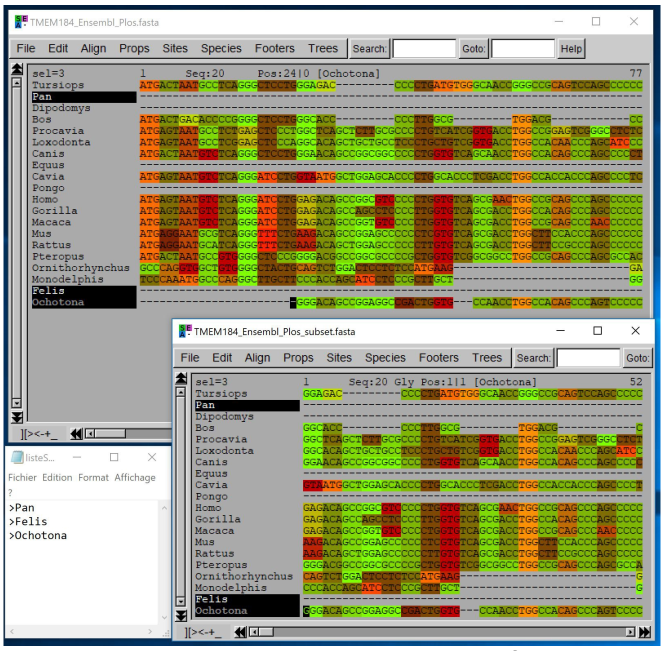
- *java -jar macse.jar -prog splitAlignment -align align.fasta -site_intervals intervals.txt*

If it is more convenient to specify the intervals to remove instead of those to keep, you can do so:

- *java -jar macse.jar -prog splitAlignment -align align.fasta -site_intervals intervals.txt -reverse_site_selection_ON*
- *java -jar macse.jar -prog splitAlignment -align align.fasta -first_site 4 -last_site 9 -reverse_site_selection_ON*

Instead of providing the interval of sites to keep, using **first_site** and **last_site** options, the limits of the interval can be automatically detected on a set of given sequences. In this case, the **first_site** (resp. **last_site**) will be the first (resp. last) site for which at least one of the given sequence has a non-gap character at this position. This is convenient when, for instance, you have one or several well annotated sequences mixed with others that may include small UTR fragments (e.g. RNAseq contigs) that you want to remove.

- *java -jar macse.jar -prog splitAlignment -align align.fasta -restrict restriction.txt*


TMEM184_Ensembl_Plos.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto: Help
sel=3 1 Seq:20 Pos:24|0 [Ochotona] 77
Tursiops
Pan
Dipodomys
Bos
Procavia
Loxodonta
Canis
Equus
Cavia
Pongo
Homo
Gorilla
Macaca
Mus
Rattus
Pteropus
Ornithorhynchus
Monodelphis
Felis
Ochotona
TMEM184_Ensembl_Plos_subset.fasta
sel=3 1 Seq:20 Gly Pos:1|1 [Ochotona] 52
listeS...
Fichier Edition Format Affichage

```
>Pan
>Felis
>Ochotona
```

Example of alignment trimming using the restrict option: the beginning of the alignment is removed until it reaches a site where at least one of the three highlighted sequences has a non-gap character

For more advanced alignment trimming options based on site content see trimAlignment subprogram.

3. Extracting an alignment containing a subset of sequences and sites

You can combine the above options to indicate simultaneously which sequences and sites to keep, e.g.:

- *java -jar macse.jar -prog splitAlignment -align TMEM184_Ensembl_Plos.fasta -subset primates.txt -first_site 679*

4. Removing sites containing only frameshifts

It is usually preferable to keep sites containing only frameshifts as they preserve the inferred reading frame. However, if you are only interested in having a good nucleotide alignment, you may prefer to consider those sites as containing only gaps and remove them:

- *java -jar macse.jar -prog splitAlignment -align align.fasta -keep_FS_OFF*

5. Spliting an alignment of amino acids

If you want to split an amino acid alignment you should specify it:

- *java -jar macse.jar -prog splitAlignment -align align.fasta -first_site 2 -last_site 3 -amino_alignment_ON*

Translate nucleotide sequences into amino acid sequences

The MACSE subprogram **translateNT2AA** translates nucleotide sequences into amino acid ones using the specified genetic codes.

1. Basic usage

The only mandatory option of this program is the **seq** option that indicates the name of the FASTA file containing the sequences to be translated:

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta*

If your input file contains aligned nucleotide sequences you can choose to ignore those gaps to obtain unaligned translated amino acids sequences (while preserving frameshifts, hence the correct reading frames):

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -ignore_gaps_ON*

Warning By default MACSE automatically removes the final stop of the sequences. This could be problematic if you plan to align amino acid sequences and to use the resulting alignment to derive an alignment of your nucleotide sequences using reportGapsAA2NT as the length of your nucleotide and amino acid sequences will no longer match. To avoid this problem you can ask MACSE to keep the final stop codon:

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -keep_final_stop_ON*

2. Ignoring some sequences or nucleotides

You can ask **translateNT2AA** to ignore sequences containing too many internal stop codons, here sequences with more than one internal stop codon will not be translated:

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -guessOneReadingFrame -maxSTOP_inSeq 0*

The default value is -1, all negative values will lead **translateNT2AA** to translate all sequences (i.e. to ignore this option).

You can also decide to remove the first and/or last codon if they are incomplete:

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -guessOneReadingFrame -trim_pending_ON*

3. Selecting the genetic code for the translation

By default translations are done using **The_Standard_Code** but you can specify a different default genetic code for your dataset using the **gc_def** option. If your dataset contains sequences that use different genetic codes you can use a text file, which contains, on separate lines, both the sequence name and the number indicating the corresponding genetic code.

All information regarding the available genetic codes and the options to indicate the code to use for translating each sequence is detailed in the genetic code section.

You can also choose to view the resulting amino acid sequences in the compressed alphabets of your choice (default is **SE_B_8**):

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -use_compressed_alphabet_ON*

4. Related documentation

You can find other options related to this program from the following links:

- genetic codes
- compressed AA alphabet
- allowed nucleotides

Alignment trimming

Because protein-coding sequence extremities may contain some UTR fragments, could be less reliable due to errors in the sequencing process, or start at different positions because of different PCR primers being used, alignment extremities are often gappy and therefore not very reliable part of alignments.

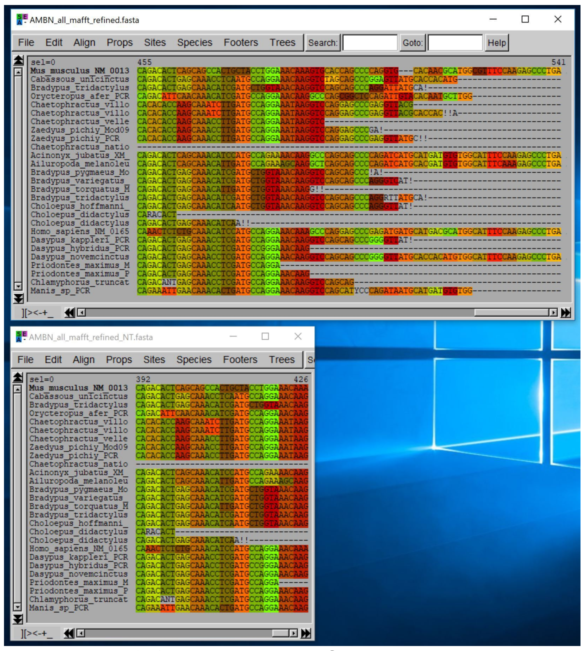
Warning This program only uses the number/fraction of gap per position to trim sites. If you want to identify internal sequence fragments that are not homologous or misaligned see trimNonHomologousFragments and reportMaskAA2NT subprograms.

1. Trimming gappy extremities of a nucleotide alignment

This subprogram allows to remove gappy positions from alignment extremities. Starting from one end of the alignment, sites are removed until a site with a large enough number (or percentage) of nucleotides is reached.

- *java -jar macse.jar -prog trimAlignment -align align.fasta -min_NT_at_ends 1*
- *java -jar macse.jar -prog trimAlignment -align AMBN_all_mafft_refined.fasta -min_percent_NT_at_ends 0.8*

The figure below displays the impact of the above trimming process on the end of the AMBN sequence alignment (a similar impact is observed at the beginning of the alignment).


AMBN_all_mafft_refined.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto: Help
AMBN_all_mafft_refined_NT.fasta
File Edit Align Props Sites Species Footers Trees

Impact of the trimming process -threshold 0.8- on the end of the AMBN alignment

In some cases, an isolated non-gappy site could stop the trimming process. To account for this possibility, you can consider the number/proportion of gaps in a sliding windows instead of considering a single site. The sliding window is centered on a site position and considers the x previous and x following sites. If the window contains enough nucleotides, the central site is the position that will determine one of the two trimming limits (one at the beginning, one at the end of the alignment). We call x the **half_window_size**.

The default trimming process considers a single site, which is equivalent to using a sliding windows of size 1 resulting from a **half_window_size** of 0.

- *java -jar macse.jar -prog trimAlignment -align align.fasta -min_percent_NT_at_ends 0.25 -half_window_size 3*

2. Trimming traceability and output statistic file

A tabular file providing information about the trimming process may be produced if required:

- *java -jar macse.jar -prog trimAlignment -align align.fasta -min_percent_NT_at_ends 0.51 -out_trim_info output_stats.csv*

Each line of this file provides information regarding the impact of the trimming process on a sequence by providing:

- **seqName**: the sequence name
- **nb_trimed_begin**: the number of nucleotides trimmed at the beginning of this sequence
- **nb_trimed_end**: the number of nucleotides trimmed at the end of this sequence
- **trimed_begin**: the fragment of the nucleotide sequence trimmed at the sequence beginning

- **trimed_end**: the fragment of the nucleotide sequence trimmed at the sequence end

3. Related documentation

You can find other options related to this program from the following links:

- allowed nucleotides
- reportMaskAA2NT: uses a NT alignment and a filtered (masked) version of its AA translation to derived the NT alignment.
- splitAlignment: splits an alignment to extract a subset of given sequences and/or sites.
- trimAlignment: trims the input alignment by removing gappy sites at the beginning/end of the alignment

Sequences trimming

URL : samples/trimSequences/

The **trimSequences** program can trim sequences by removing stops, deletions, insertions and frame shifts :

- *java -jar macse.jar -prog trimSequences -align align.fasta -seq sequences.fasta*

Note that sequences missing from **align.fasta** will be added to it.

1. Nucleotides, trimmed sequences, annotations and statistics outputs

a) Nucleotides output

URL : samples/trimSequences/out_NT/

You can redefine the nucleotides sequences output file path/name with the **out_NT** option :

- *java -jar macse.jar -prog trimSequences -align align.fasta -seq sequences.fasta -out_NT output_NT.fasta*

Without this option, the file path/name is based on alignment file path/name (**align_NT.fasta**).

b) Trimmed sequences output

URL : samples/trimSequences/out_NT_trimmed/

The trimmed sequences output file path/name can also be redefined :

- *java -jar macse.jar -prog trimSequences -align align.fasta -seq sequences.fasta -out_NT_trimmed output_trimmed.fasta*

By default, if this option is not used, MACSE will use the alignment file path/name (**align_trimmed.fasta**).

c) Annotations output

URL : samples/trimSequences/out_NT_annotated/

MACSE will output a file with lower case and upper case letters to show what alignment parts were trimmed :

- *java -jar macse.jar -prog trimSequences -align align.fasta -seq sequences.fasta -out_NT_annotated output_annotated.fasta*

Default file path/name depends on alignment file path/name (**align_annotated.fasta**).

d) Statistics output

URL : samples/trimSequences/out_trim_stat/

A statistics file is created after each trimming and contains the following data :

- *Sequence name*
- *Trimmed prefix length*
- *Conserved acids length*
- *Trimmed suffix length*
- *Total acids length*

The path/name of the statistics file can be redefined with the **out_trim_info** option :

- *java -jar macse.jar -prog trimSequences -align align.fasta -seq sequences.fasta -out_trim_info output_stats.csv*

Without this option, MACSE will base on alignment file path/name (**align_stats.csv**).

2. Less reliable sequences

URL : samples/trimSequences/seq_lr/

This program can also manage reliable and less reliable sequences with **seq** and **seq_lr** options :

- *java -jar macse.jar -prog trimSequences -align align.fasta -seq sequences.fasta -seq_lr sequences_lr.fasta*

3. Other options

You can find other options related to this program in the following links :

- costs
- nucleotides

Triming non-homologous fragments before alignment

The **TrimNonHomologousFragments** subprogram was developed to specifically filter long insertions that often result from annotation errors of introns introduced in CDSs or from alternative splicing. Indeed, positioning long insertions in one or several sequences could drastically slow down the alignment process. Long indels may olaso often prove finally useless since they are removed by alignment filtering tools in subsequent analyses. Finally, non-homologous fragments are often not correctly aligned as indels (see example below) and would hence drastically influence the inferred dN/dS values.

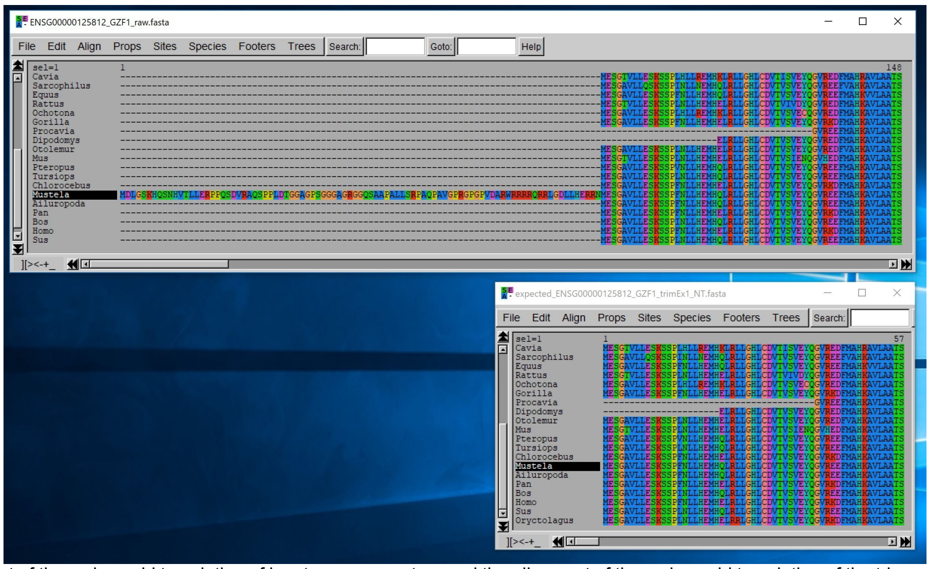
When a compatibility graph of Maximum Exact Match (MEM) is constructed between two genomic sequences, they can be aligned after identification of the longest weighted path Hohl et al. 2002. We extended this approach to handle the translation of nucleotide sequences in the three possible coding frames using a compressed amino acid alphabet. This allowed identifying and suppressing long insertions present in only few sequences before sequence alignment, as such regions, are part of few optimal, or nearly optimal, MEM paths.

Warning This programs mainly aims at removing long non-homologous fragments. Smaller ones will be kept, to avoid removing fragments that are really homologous at this step. If your analyses are sensitive to such non-homologous fragments (e.g. dN/dS estimation), we strongly advice to also use a post filtering of your alignment at the amino acid level (e.g. using HMMCleaner, BMGE or trimAl) and to report this AA masking/filtering at the nucleotide level using reportMaskAA2NT.

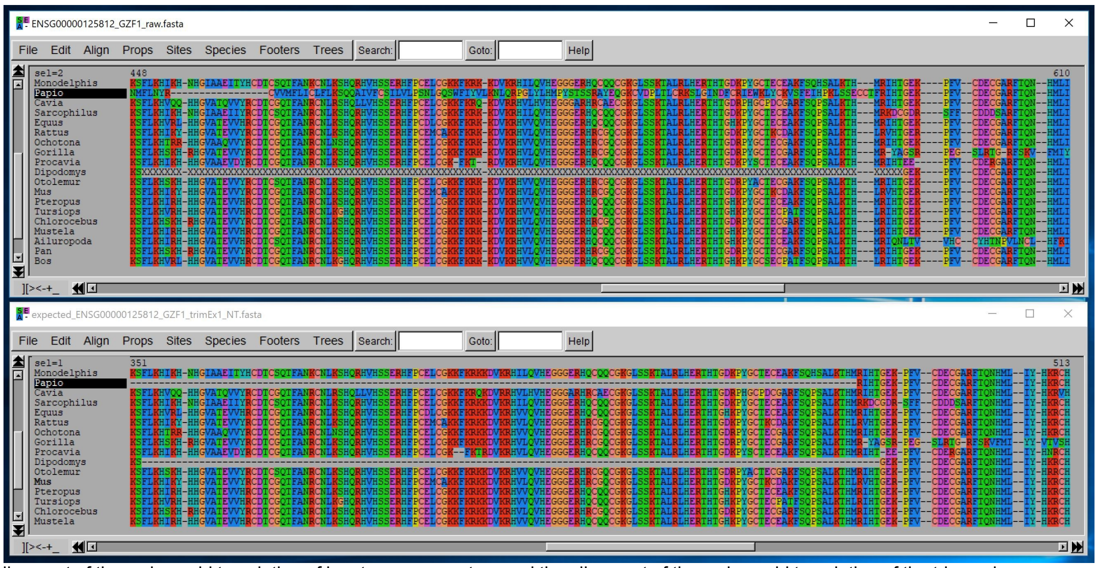
1. Simple trimming of non-homologous sequence fragments

- *java -jar macse.jar -prog trimNonHomologousFragments -seq ENSG00000125812_GZF1_raw.fasta -out_trim_info output_stats.csv*

Although this trimming is done on unaligned nucleotide sequences and returns unaligned nucleotide sequences, we display below the aligned translation of those sequences (using MUSCLE) to illustrate the relevance of this process:


ENSG00000125812_GZF1_raw.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto: Help
sel=1
Cavia
Sarcophilus
Equus
Rattus
Ochotona
Gorilla
Procavia
Dipodomys
Otolemur
Mus
Pteropus
Tursiops
Chlorocebus
Mustela
Ailuropoda
Pan
Bos
Homo
Sus
expected_ENSG00000125812_GZF1_trimEx1_NT.fasta
Oryctolagus

Beginning of the alignment of the amino acid translation of input sequences -top- and the alignment of the amino acid translation of the trimmed sequences -bottom-

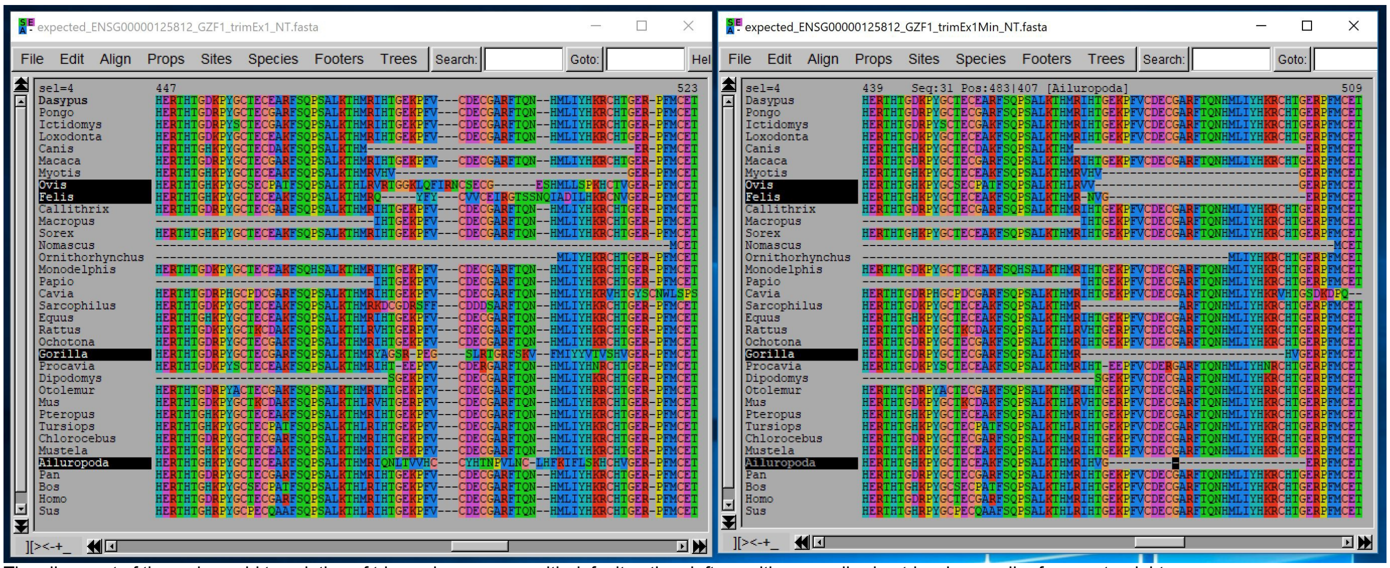


Middle of the alignment of the amino acid translation of input sequences -top- and the alignment of the amino acid translation of the trimmed sequences -bottom-

You can control this filtering using different options, but the key ones are the minimal length that a non-homologous fragment should have to be removed when it occurs within a sequence, and when it occurs at a sequence extremity.

- *java -jar macse.jar -prog trimNonHomologousFragments -seq ENSG00000125812_GZF1_raw.fasta -out_trim_info output_stats.csv -min_trim_in 40 -min_trim_ext 20*

The result of this trimming is compared with the trimming obtained using default options based on the alignment of their amino acid sequences:


expected_ENSG00000125812_GZF1_trimEx1_NT.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto: Hel
expected_ENSG00000125812_GZF1_trimEx1Min_NT.fasta
File Edit Align Props Sites Species Footers Trees Search: Goto:
sel=4
Seq:31 Pos:483|407 [Ailuropoda]
Dasypus
Pongo
Ictidomys
Loxodonta
Canis
Macaca
Myotis
Ovis
Felis
Callithrix
Macropus
Sorex
Nomascus
Ornithorhynchus
Monodelphis
Papio
Cavia
Sarcophilus
Equus
Rattus
Ochotona
Gorilla
Procavia
Dipodomys
Otolemur
Mus
Pteropus
Tursiops
Chlorocebus
Mustela
Ailuropoda
Pan
Bos
Homo
Sus
The alignment of the amino acid translation of trimmed sequences with default option -left- or with some allowing trimming smaller fragments -right-

2. Removing sequences that are non-homologous and understanding the output statistic file

If most sites in a sequence are trimmed in this the process, this sequence is likely not homologous to others and it could be better to remove it. You can decide which fraction of the sequence should remain after trimming for a sequence to be kept in the output fasta file:

- *java -jar macse.jar -prog trimNonHomologousFragments -seq ENSG00000125812_GZF1_raw.fasta -out_trim_info output_stats.csv -min_homology_to_keep_seq 0.6 -min_trim_in 40 -min_trim_ext 20*

A tabular file providing information about the trimming process is provided with one line per sequence and the following columns:

- seqName: the sequence name
- initialSeqLength: the initial sequence length (before trimming)
- nbKeep: the number of nucleotides/characters that remains after trimming
- nbTrim: the number of nucleotides/characters that have been removed by the trimming process (including non informative nucleotides 'N')
- nbInformativeTrim: the number of informative nucleotides/characters that have been removed by the trimming process (excluding non informative nucleotides 'N')
- percentHomologExcludingExtremities: once extremities have been trimmed which fraction of the remaining part of the sequence has also been trimmed
- percentHomologIncludingExtremities: which fraction of the sequence has also been trimmed
- keptSequences: is the sequence kept and included in the output fasta file

You can specify the name of this output CSV file:

- *java -jar macse.jar -prog trimNonHomologousFragments -seq sequences.fasta -out_trim_info output_stats.csv*

3. Trimming gappy alignment extremities

To trim only alignment extremities and remove gappy alignment extremities you can use exportAlignment or trimAlignment.

4. Related documentation

You can find other options related to this program from the following links:

- genetic codes
- alignment costs
- compressed AA alphabet
- allowed nucleotides

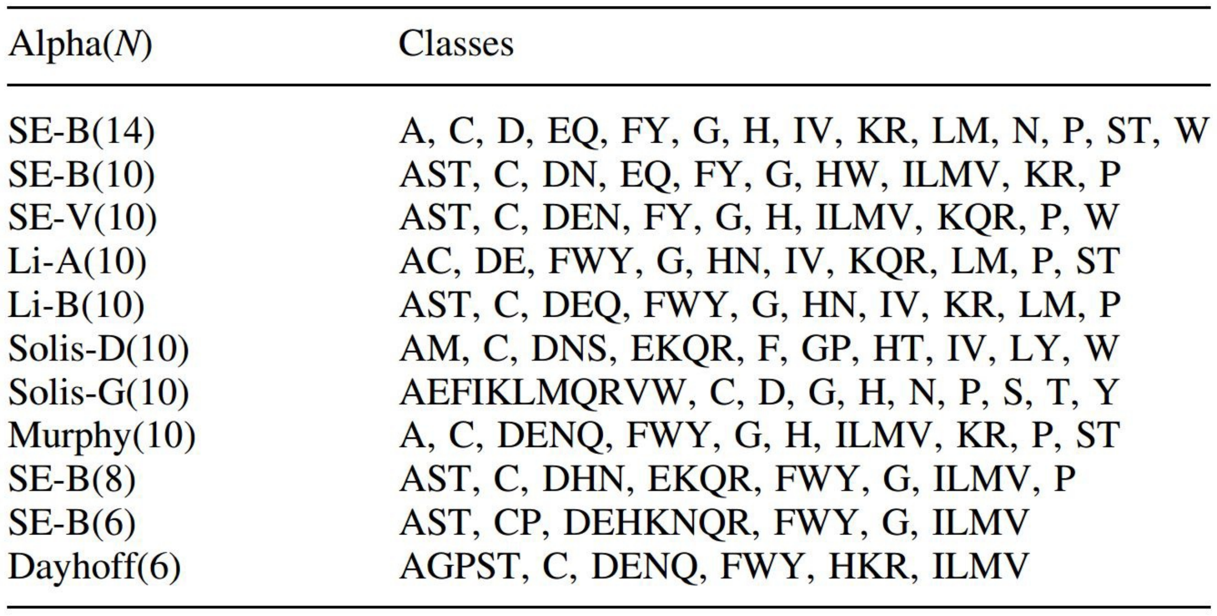
Compressed amino acid alphabet

Folder: samples/delegations/alphabet/

In some MACSE subprograms, you can use the **alphabet_AA** option to change the default compressed amino acid alphabet used by MACSE to compute initial pairwise distances or to identify homologous fragments. For instance the command to select the **Dayhoff_6** alphabet is:

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -use_compressed_alphabet_ON -alphabet_AA Dayhoff_6*

The proposed alphabets are those listed in the Edgar 2004 paper.



Alpha(*N*)	Classes
SE-B(14)	A, C, D, EQ, FY, G, H, IV, KR, LM, N, P, ST, W
SE-B(10)	AST, C, DN, EQ, FY, G, HW, ILMV, KR, P
SE-V(10)	AST, C, DEN, FY, G, H, ILMV, KQR, P, W
Li-A(10)	AC, DE, FWY, G, HN, IV, KQR, LM, P, ST
Li-B(10)	AST, C, DEQ, FWY, G, HN, IV, KR, LM, P
Solis-D(10)	AM, C, DNS, EKQR, F, GP, HT, IV, LY, W
Solis-G(10)	AEFIKLMQRVW, C, D, G, H, N, P, S, T, Y
Murphy(10)	A, C, DENQ, FWY, G, H, ILMV, KR, P, ST
SE-B(8)	AST, C, DHN, EKQR, FWY, G, ILMV, P
SE-B(6)	AST, CP, DEHKNQR, FWY, G, ILMV
Dayhoff(6)	AGPST, C, DENQ, FWY, HKR, ILMV

Details on compressed alphabet, source Edgar 2004, NAR

Genetic codes and translation

1. Selecting the adapted genetic code(s)

Folder: samples/delegations/aminos/gc_def/

MACSE is able to handle the following genetic codes:

- 1 *The_Standard_Code*
- 2 *The_Vertebrate_Mitochondrial_Code*
- 3 *The_Yeast_Mitochondrial_Code*
- 4 *The_Mold_Protozoan_and_Coelenterate_Mitochondrial_Code_and_the_Mycoplasma_Spiroplasma_Code*
- 5 *The_Invertebrate_Mitochondrial_Code*
- 6 *The_Ciliate_Dasycladacean_and_Hexamita_Nuclear_Code*
- 9 *The_Echinoderm_and_Flatworm_Mitochondrial_Code*
- 10 *The_Euplotid_Nuclear_Code*
- 11 *The_Bacterial_Archaeal_and_Plant_Plastid_Code*
- 12 *The_Alternative_Yeast_Nuclear_Code*
- 13 *The_Ascidian_Mitochondrial_Code*
- 14 *The_Alternative_Flatworm_Mitochondrial_Code*
- 15 *Blepharisma_Nuclear_Code*
- 16 *Chlorophycean_Mitochondrial_Code*
- 21 *Trematode_Mitochondrial_Code*
- 22 *Scenedesmus_obliquus_mitochondrial_Code*
- 23 *Thraustochytrium_Mitochondrial_Code*

By default MACSE uses **The_Standard_Code** but you can specify a different default genetic code for your dataset using the **gc_def** option, specifying the number of the genetic code (listed above):

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -gc_def 9*

For instance, to align grasshoper COI genes and pseudogenes (data from Song et al 2008), the correct genetic code is the invertebrate mitochondrial one (5):

- *java -jar macse.jar -prog alignSequences -seq Song_PNAS2008_Grasshoppers_COX1_genes.fasta -seq_lr Song_PNAS2008_Grasshoppers_COX1_pseudo.fasta -gc_def 5 -stop_lr 30 -fs_lr 30*

If your dataset contains sequences that use different genetic codes you can use a text file, which on each line contains both the sequence name and the number of the corresponding genetic code (you can use any of the following field separator: space, tabulation, comma, semicolon):

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -gc_file riboSequences.txt*

Any sequence absent from this file will be translated using default genetic code (see **gc_def** option).

2. Translation of codon with ambiguous nucleotides

Folder: samples/delegations/aminos/ambi/

When a codon contains a single ambiguous nucleotide (e.g. N, R, Y) and all its possible translation lead to the same amino acid, MACSE uses this unambiguous translation despite the nucleotide ambiguity. You can disable this functionality, to translate any codon containing an ambiguous nucleotide into the unknown amino acid (X) using the **ambi_OFF** option. In this case, with the default genetic code, the codon **TCN** will be translated into the unknown amino acid (**X**) instead of being translated into serine (**S**).

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -ambi_OFF*

Alignment cost parameters

MACSE defines the score/cost of a nucleotide alignment based on its amino acid translation. The cost of an alignment depends on the elementary costs of the events it encompassed:

- the bonus/cost of one amino acid facing another is defined in the amino acid substitution matrix (default or specified, cf 7:alternative substitution matrix)
- the cost of a stretch of *n* consecutive gaps is defined as 1/ a fix cost for opening this gap stretch plus 2/ an extension cost proportional to its length *n*
- the cost of a frameshift and a stop codon may vary depending on 1/ whether they are inside the sequence or at its extremities and 2/ the type of sequence where they appear (reliable or not, see **seq_lr** option example)

MACSE provides some default values, but you can set all those costs at another value. Note that:

- if you enter a positive value for a cost it will automatically be converted into negative values (you do not need to worry about signs).
- you can use up to one decimal for costs.

1. names of the cost options

- cost options related to gap, frameshift and stop codon have names starting by **gap**, **FS** and **stop** respectively.
- gap cost options related to opening or extension have **op** or **ext** in their names
- cost options specific to less reliable sequences have **lr** in their names
- penalties for events appearing at the end(s) of a sequence have **term** in their names
- *term* stands for terminal and refers to the end of a sequence, indifferently to the beginning or the end for gaps and frameshifts, but only to the end of the sequence for stop codons.

2. specifying gap costs and fixing unexpected results

- gaps are equally penalized in all sequences, reliable (*-seq*) or not (*-seq_lr*).
- usually gaps at the beginning or at the end of the alignment are less penalized as they can simply result from the fact that the locus is not completely sequenced (different primer set or shorter contig).

In MACSE options, gaps located between the first and the last amino acid are called **internal** gaps whereas those before the first amino acids or after the last one are called **terminal** gaps.

Folder: samples/delegations/costs/

You can adjust all or some gap options using the following options:

for gap opening within the sequence:

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -gap_op 15*

for terminal gaps opening (gaps not within the sequence):

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -gap_op_term 20*

for gaps extension within a sequence

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -gap_ext 10*

for terminal gaps extension (gaps not within the sequence):

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -gap_ext_term 10*

Some guidelines for gaps:

- If you have too many small gaps, you can try to increase the penalty of gap opening (this would favor alignment with fewer but longer gaps).
- If you have some incomplete sequences you can try to decrease the penalty of external gaps.
- There is unfortunately no perfect way to fix those costs and this is one of the key challenges of multiple sequence alignment software.

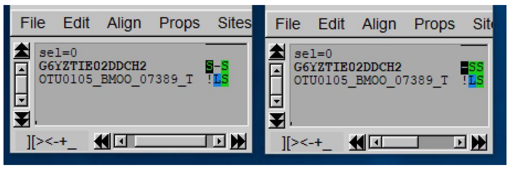
Even with a very simple dataset, you can have results that seems unexpected. By default MACSE penalizes only slightly more internal gaps than terminal ones. This could produce that seems unexpected as the one below (Folder:samples/delegations/costs/unexpected/):

- *java -jar macse.jar -prog alignSequences -seq small_weird.fasta*

A very different result could be obtained by penalizing less external gaps:

- *java -jar macse.jar -prog alignSequences -seq small_weird.fasta -gap_op 7 -gap_op_term 3.5*

The figure below displays the alignment obtained with default costs (on the left) and the one obtained when penalizing less external gap opening (on the right). The alignment on the right has a penalty of -4 due to the L (leucine) facing the S (Serine) that the one of the right do not have by default the bonus of having an internal gap instead of an external gap is not enough to compensate this -4 penalty, and the alignment on the left is favor (with a slight score difference -14 vs -14.3).



```
sel=0
G6YZTIE02DDCH2        S-S
OTU0105_BMOO_07389_T  !LS
```

```
sel=0
G6YZTIE02DDCH2        -SS
OTU0105_BMOO_07389_T  !LS
```

3. Specifying frameshift cost

Frameshift can be seen as an unexpected/unlikely kind of gap stretch (a nucleotide gap stretch that is not a multiple of 3 in a coding nucleotide sequence). They could be real biological events (e.g. in pseudogenes, or in sequence that will be post-processed by the molecular machinery) or due to sequencing/annotation errors. While MACSE allows such events, they are strongly penalized in reliable sequences, as they are unexpected in nucleotide coding sequences, and a little less in less reliable sequences.

By default, a frameshift in the first or last codon is not penalized by MACSE as this usually reflects an incomplete sequence or a UTR annotation error that are quite frequent.

Folder: samples/delegations/costs/fs/

You can specify **4 different frameshift costs** for internal vs external frameshifts, appearing in reliable vs less reliable sequences:

Internal frameshifts in *reliable* sequences

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -fs 5*

Terminal frameshifts in *reliable* sequences

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -fs_term 30*

Internal frameshifts in *less reliable* sequences

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -seq_lr sequences_lr.fasta -fs_lr 15*

Terminal frameshifts in *less reliable* sequences

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -seq_lr sequences_lr.fasta -fs_lr_term 20*

4. Internal stops

By default stop codons are not penalized at the end of a sequence; they are penalized everywhere else.

As stop codons are expected at the end of the sequences their cost is always null. You can only specify **2 different stop costs** for internal stops appearing in reliable vs less reliable sequences:

specifying cost for stop condons within *reliable sequences*

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -stop 10*

specifying cost for stop condons within *less reliable sequences*

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -seq_lr sequences_lr.fasta -stop_lr 40*

Warning concerning FS and stop costs:

For a given sequence type a stop codon should never cost more than two frameshifts, otherwise all stop codons will be avoided by MACSE by including two successive frameshifts.

5. Aligning functional and non-functional sequences (pseudogenes).

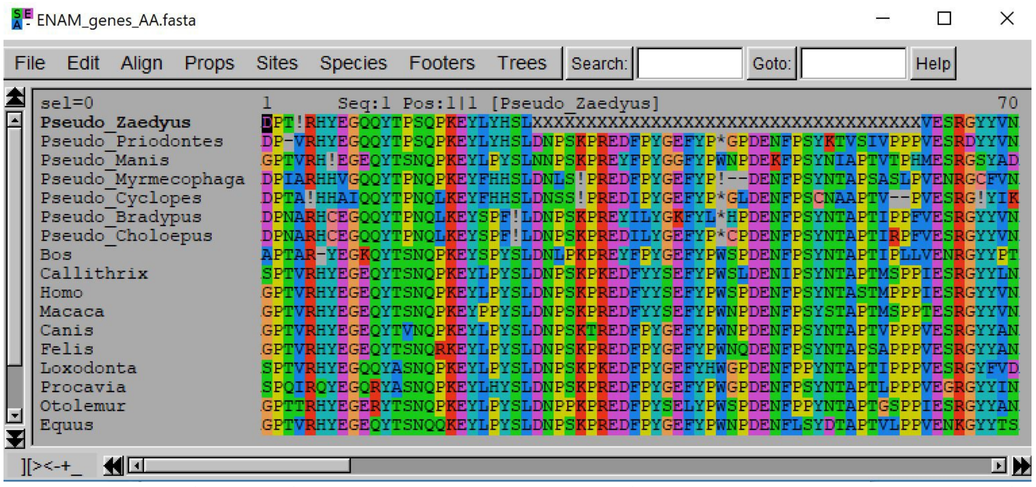
If you know, a priori, which sequences are functional protein-coding sequences and which ones are non-functional (i.e.

pseudogenes), it is preferable to have them in different files (as described in 'Sequence alignment 3. Less reliable sequences'). You can then ask MACSE to use different costs for each file, assigning standard costs to the file containg functional protein-coding sequences (option **fs** and **stop**) and lower costs for the other file, containing the less reliable sequences (option **fs_lr** and **stop_lr**).

In **pseudogenes** a STOP can result from a single mutation whereas frameshifts require one insertion/deletion. It thus seems reasonable to not penalize stop codons more than frameshifts in such sequences, e.g. using a penalty between 10 and 30 for both events.

For pseudo-genes **-fs_lr 10 -stop_lr 10** is often a good initial parameter set.

- *java -jar macse.jar -prog alignSequences -seq ENAM_genes.fasta -seq_lr ENAM_pseudos.fasta -fs_lr 10 -stop_lr 10*



```
sel=0               1      Seq:1 Pos:1|1 [Pseudo_Zaedyus]                      70
Pseudo_Zaedyus      DPT!RHYEGQQYTPSQPKEYLYHSLXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXVESRGYYVN
Pseudo_Priodontes   DP-VRHYEGQQYTPSQPKEYLYHSLDNPSKPREDFPYGEFYP*GPDENFPSYKTVSIVPPPVESRDYYVN
Pseudo_Manis        GPTVRH!EGEQYTSNQPKEYLPYSLNNPSKPREYFPYGGFYPWNPDEKFPSYNIAPTVTPHMESRGSYAD
Pseudo_Myrmecophaga DPIARHHVGQQYTPNQPKEYFHHSLDNLS!PREDFPYGEFYP!--DENFPSYNTAPSASLPVENRGCFVN
Pseudo_Cyclopes     DPTA!HHAIQQYTPNQLKEYFHHSLDNSS!PREDIPYGEFYP*GLDENFPSCNAAPTV--PVESRG!YIK
Pseudo_Bradypus     DPNARHCEGQQYTPNQLKEYSPF!LDNPSKPREYILYGKFYL*HPDENFPSYNTAPTIPPFVESRGYYVN
Pseudo_Choloepus    DPNARHCEGQQYTPNQLKEYSPF!LDNPSKPREDILYGEFYP*CPDENFPSYNTAPTIRPFVESRGYYVN
Bos                 APTAR-YEGKQYTSNQPKEYSPYSLDNLPKPREYFPYGEFYPWSPDENFPSYNTAPTIPLLVENRGYYPT
Callithrix          SPTVRHYEGEQYTSNQPKEYLPYSLDNPSKPKEDFYYSEFYPWSLDENIPSYNTAPTMSPPIESRGYYLN
Homo                GPTVRHYEGEQYTSNQPKEYLPYSLDNPSKPREDFYYSEFYPWSPDENFPSYNTASTMPPPIESRGYYVN
Macaca              GPTVRHYEGEQYTSNQPKEYPPYSLDNPSKPREDFYYSEFYPWNPDENFPSYSTAPTMSPPTESRGYYVN
Canis               GPTVRHYEGEQYTVNQPKEYLPYSLDNPSKTREDFPYGEFYPWNPDENFPSYNTAPTVPPPVESRGYYAN
Felis               GPTVRHYEGEQYTSNQRKEYLPYSLDNPSKPREDFPYGEFYPWNQDENFPSYNTAPSAPPPVESRGYYAN
Loxodonta           SPTVRHYEGQQYASNQPKEYLPYSLDNPSKPKEDFPYGEFYHWGPDENFPPYNTAPTIPPPVESRGYFVD
Procavia            SPQIRQYEGQRYASNQPKEYLHYSLDNPSKPREDFPYGEFYPWGPDENFPSYNTAPTLPPPVEGRGYYIN
Otolemur            GPTTRHYEGERYTSNQPKEYLPYSLDNPPKPREDFPYSELYPWSPDENFPPYNTAPTGSPPIESRGYYAN
Equus               GPTVRHYEGEQYTSNQQKEYLPYSLDNPSKPREDFPYGEFYPWNPDENFLSYDTAPTVLPPVENKGYYTS
```

Note that in this alignment, frameshifts and stop codons are quite frequent and correctly recovered for pseudogenes but are rare for others.

Note that you can test other values (e.g. between 10 and 30) since the frequency of frameshifts ans stops in your pseudogene sequences (hence the relevant cost) is related to the time elapsed since your sequences are no longer coding.

- *java -jar macse.jar -prog alignSequences -seq Song_PNAS2008_Grasshoppers_COX1_genes.fasta -seq_lr Song_PNAS2008_Grasshoppers_COX1_pseudo.fasta -gc_def 5 -stop_lr 30 -fs_lr 30*

6. Aligning NGS data (reads or contigs).

When aligning a set of sequences containing NGS data it is preferable, as for pseudogenes, to split the sequences in two files and to use different costs for frameshifts and stop codons in order to account for potential sequencing errors in **NGS reads or contigs**. For NGS raw sequences (reads or contigs) **-fs_lr 10 -stop_lr 15** is often a good initial parameter set. **You can of course adapt these costs depending on the sequencing technology used** since some technologies induce more frequent errors in homopolymers (e.g. 454) seen as frameshifts by MACSE, while others rather induce nucleotide calling errors leading to potential stop codons. The ideal case is to have a set or reliable sequences (i.e. from public databases or manually annotated) that you can provide as reference to MACSE in order to guide the alignment process. In this exemple, we identified, by BLAST, reads similar to Ensembl sequences ENSG00000119777 (TMEM214) and used orthologous CDS sequences to guide the alignments of 454 contigs while accounting for probable sequencing errors:

- *java -jar macse.jar -prog alignSequences -seq TMEM214.fasta -seq_lr TMEM214_reads.fasta -fs_lr 10 -stop_lr 15*

Note that the more reliable sequences the better. If you include numerous reliable sequences or have a clean alignment corresponding to your current contigs, you can use the enrichAlignment to rapidly add new contigs to your initial alignment (see corresponding sections for detailed examples).

7. Using an alternative substitution matrix

Folder: samples/delegations/costs/scoreMatrix/

- *java -jar macse.jar -prog alignSequences -seq sequences.fasta -score_matrix VTML240*

Available substitution matrix are:

- *BLOSUM62*
- *VTML200*BIS
- *VTML240*

Note that all default costs are set for the Blosum62 matrix. If you decide to use a different matrix you should probably use different costs for all other options.

Nucleotides

Unidentified nucleotides

URL : samples/delegations/nucleos/allow_NT/

Nucleotides alignments may contain inexistent nucleotides which can prevent MACSE from working correctly, you can bypass this problem using the **allow_NT** option :

- *java -jar macse.jar -prog translateNT2AA -seq sequences.fasta -allow_NT "?"*

This example will read **allow_NT** nucleotides as if they were 'N' letters.

Note that some characters can be misinterpreted by the console, so it is advised to add double quotes.