

trimAl: a tool for automated alignment trimming in large-scale phylogenetics analyses

Salvador Capella-Gutiérrez, Jose M. Silla-Martínez and Toni Gabaldón

Version 1.2b

Index of contents

1. General features of trimAl.

1.1 Input and output formats.	3
1.2 Algorithm parameters.	3
1.2.1 Gap Score.	3
1.2.2 Residue Similarity Score.	4
1.2.3 Identity Score.	5
1.2.4 Consistency Score.	5
1.2.5 Residue Overlap Score.	5
1.2.6 Sequence Overlap Score.	5
1.2.7 Window Size.	5
1.3 Trimming algorithms and heuristics.	6
1.3.1 Removal of manually selected columns (-selected).	6
1.3.2 Alignment trimming based on user-defined thresholds.	6
1.3.3 Alignment selection and trimming based on consistency thresholds.	7
1.3.4 Alignment trimming based on automatically-selected thresholds.	7
1.3.4.1 The -gappyout method.	7
1.3.4.2 The -strict method.	8
1.3.4.3 The -strictplus method.	10
1.3.5 Automatic trimming heuristic: automated1.	11
1.3.6 Automated removal of spurious sequences.	12

1. General features of trimAl.

Here we provide a summarized but comprehensive documentation of the main features of *trimAl* v1.2. More details, additional information, as well as specific examples to illustrate the use of the software can be found at trimAl website: trimal.cgenomics.org

1.1 Input and output formats.

trimAl reads and renders protein or nucleotide alignments in several Multiple Sequence Alignment (MSA) formats, including Phylip, Fasta, Clustal, NBRF/Pir, Mega and Nexus. The program detects automatically the input format and generates the output file (a trimmed alignment according to the options selected) in the same format. Alternatively, the user can select a different format for the output. Moreover, *trimAl* can provide as an output the complementary MSA, that is, the columns that would otherwise be removed by the specified parameters (option *-complementary*). Finally, to facilitate the visualization of trimAl's trimming, the program can generate an html file in which selected and trimmed columns are colored differently (*-htmlout*).

Besides MSAs, *trimAl* can optionally produce outputs other type of MSAs, which have been deemed of interest. For instance, to facilitate the tracking of the correspondences between the columns in the original and the trimmed alignment, trimAl can return the relationship between their column numbers (option *-columnnumbering*). *trimAl* can provide information on gap and/or conservation scores in a MSA. This information can be relative to each column, options *-sgc* for gaps and *-scc* for conservation values, or it can show the distribution of values along the alignment, options *-sgt* and *-sct* for gaps and conservation distribution, respectively. When comparing several alignments, *trimAl* can also offer statistical information about their consistency score (options *-sfc* for each column and *-sft* for the whole alignment). Finally, *trimAl* can provide a comparison matrix summarizing the percentage of identities between each pair of sequences in the alignment, their averages and the highest identity pair for each sequence, option *-sident*.

1.2 Algorithm parameters

In order to apply the different trimming algorithms and heuristics (see next section), *trimAl* can compute a number of different scores, which are defined below.

1.2.1 Gap Score (Sg).

The gap score for a column of size n is the fraction of positions in the column without a gap.

$$Sg(k) = 1.0 - (\text{number of sequences with a gap} / n)$$

1.2.2 Residue Similarity Score.

The residue similarity score consists of Mean Distance (MD) scores as described in Thompson et al, (2001). This value uses the score between any pair of residues a and b , $C(a,b)$, as defined by a given scoring matrix. By default, trimAl uses the BLOSUM62 matrix to work with amino acids sequences and the identity matrix with nucleotides residues, but other matrices can also be supplied by the user.

The similarity score of a column is computed as follows. Given a scoring matrix (e.g. BLOSUM62) with R residues, an R -dimensional continuous sequence space is defined. For each residue i in the column k , a point S_i in the space is defined as:

$$S_i = \begin{pmatrix} C(1, A_{ik}) \\ C(2, A_{ik}) \\ \vdots \\ C(R, A_{ik}) \end{pmatrix}$$

Being A_{ik} the residue i -th of the column k and $C(1, A_{ik})$ the substitution score in the residue similarity matrix between the first residue in the alphabet and A_{ik} . In other words, S_i is just a column of the scoring matrix selected by the residue in the sequence i -th of the column k . S_{ir} is defined as the r -th dimension of the point S_i .

Then, for column k , the distance D_{ij} for each pair of sequences i and j is simply their euclidean distance in the r -dimensional space defined above:

$$D_{ij} = \sqrt{\sum_{r=1}^R (S_{ir} - S_{jr})^2}$$

In order to give more weight in the final score to distantly related sequence pairs a weighted mean of the pairwise sequence distances D_{ij} is calculated as follows:

$$Q = \frac{\sum_{j=i+1}^N \sum_{i=1}^N W_{ij} D_{ij}}{\sum_{j=i+1}^N \sum_{i=1}^N W_{ij}}$$

Where W_{ij} is defined as $100.0 - PCID_{ij}$, being PCID the percentage of sequence identity between sequences i and j . After that, the similarity score of the column k is defined as:

$$MD_k = \exp(-Q)$$

In *trimAl* v1.2, if the column k has a gap score equal or lower than 0.2, the MD_k score is set to zero. This is a way to penalize the presence of many gaps in the column in order to avoid that columns with few residues receive high scores. We noticed that the inclusion of this penalty avoided clear pitfalls when the trimming of an alignment was performed using only similarity information.

1.2.3 Identity Score.

The identity score for each possible pair of sequences in the alignment is the number of identical residues aligned between these two sequences divided by the length of the longer sequence.

1.2.4 Consistency Score.

The column consistency score can only be computed when more than one alignment for the same set and in the same order of sequences is provided.

Each alignment, called reference alignment when selected, is compared with the rest of the alignments. Then each pair of aligned residues in the reference alignment is compared with the other alignments and 1.0 is added to the cumulative score every time the same aligned residue pair is found in one of the other alignments. This cumulative score is then divided by the total number of alignments considered and by the total number of pairs in the reference alignment; so that the final score ranges from 0 (no aligned pair found in the other alignments) to 1 (all alignments have the same pairs and are thus fully consistent).

1.2.5 Residue Overlap Score.

To calculate this score *trimAl* considers only three types of elements, namely "gap", "residue" or "indetermination". The overlap score of a column in a given sequence is calculated as the number of times that the program finds the same element at the same position in the other sequences of the alignment, divided by the size of this subset.

1.2.6 Sequence Overlap Score.

After defining a residue overlap (see above) threshold, set by the user, we define the sequence overlap score as the percentage of residues in that sequence that pass that threshold.

1.2.7 Window Size.

This value establishes the number of columns at each side of a given position that *trimAl* has to consider when computing some scores, such as gap, conservation or consistency scores, for that position. When a window size is given, *trimAl* provides the average value of all columns considered.

1.3 Trimming algorithms and heuristics

1.3.1 Removal of manually selected columns (-select).

This algorithm simply removes a set of columns as indicated by the user. The set of columns that will be removed has to be provided as individual column numbers separated by commas, and/or blocks of consecutive columns indicated as the first and last column number separated by hyphen. In the following example:

```
-select {n,l,m-k}
```

where *n* and *l* are interpreted as single column numbers while *m-k* is a range of columns (from column *m* to column *k*, both included) to be deleted. Note that the numbering of the columns starts by 0.

For instance, the command:

```
-select {2,7,20-25,80-100}
```

will remove columns 2 and 7 and two blocks of columns ranging from column 20 to 25 and 80 to 100, respectively.

1.3.2 Alignment trimming based on user-defined thresholds.

The user can choose to remove all columns that do not pass a given threshold or a combination of thresholds. The gap threshold (*-gt*) and similarity threshold (*-st*) correspond to minimal values of the respective scores explained above and can be used alone or in combination. As the scores they refer to, both thresholds range from 0 to 1.

trimAl provides two shortcuts to widely used thresholds: *-nogaps* (equivalent to *-gt 1*), that deletes all columns with at least one gap in it, and *-noallgaps*, which removes those columns composed only by gaps.

Additionally, the user can set up a conservation threshold (*-cons*) which refers to the minimum percentage of columns from the input alignment that should be part of the trimmed alignment. This threshold is defined between 0 and 100. This threshold overrides all other thresholds. That is, if any other threshold would render a trimmed alignment with fewer columns than those stated by the conservation threshold, then more columns are added to the trimmed alignment until the conservation threshold is fulfilled. These columns are added in the order dictated by their scores, always adding first the columns with the highest scores. In the case *trimAl* has to decide upon columns with equal scores, then columns adjacent to already selected column-blocks and closer to the center of the alignment are added first, prioritizing the extension of longer and central blocks.

1.3.3 Alignment selection and trimming based on consistency thresholds.

When a set of Multiple Sequence Alignments is provided, *trimAl* computes a consistency score for each alignment in the set and, subsequently, the alignment with the highest score is then chosen.

The selected alignment might be trimmed in different ways. One of them is by removing the columns that are less consistent across the other alignments. In order to do that, the user can use the (-ct) parameter to set up the minimal values of the consistency score (range of values between 0 and 1). All columns that do not reach this value will be removed. The conservation score can also be used here as explained above or, alternatively, in combination with gap or/and similarity methods.

1.3.4 Alignment trimming based on automatically-selected thresholds.

trimAl has different automatic methods to select different thresholds depending on MSA features; The gappyout, strict and strictplus methods, which are described below.

1.3.4.1 The -gappyout method.

This method is based on the MSA's gap distribution. In a first step, this method computes the gap scores of all columns and sorts them according to this score, producing a plot of possible gap score thresholds versus the percentage of the alignment below that threshold (see figure S1). Subsequently, for each set of three consecutive points in this plot *trimAl* computes the slopes between the first and third point (blue lines). After comparing all slopes, *trimAl* selects the point of maximal variation between consecutive slopes (vertical red line in the figure S1).

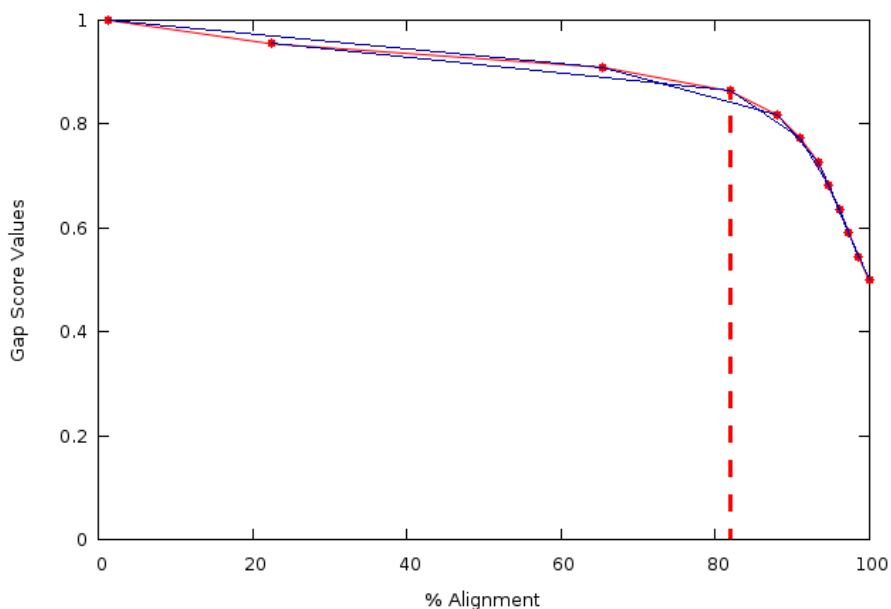


Figure S1. Example of an internal *trimAl* plot showing possible gap score thresholds (y axis) versus percentages of alignment length below that threshold (x axis). Thin blue lines

indicate slopes computed by the program. The vertical blue line indicates the cut-off point selected by the gappyout algorithm.

After the selection of a gap score cut-off point, *trimAl* removes all of those columns that do not reach this value. In practice, this method basically detects the bimodal distribution of gap scores (gap rich and gap poor columns) in an alignment to subsequently get rid of the gap-rich mode. In our benchmarks, we have observed that this method efficiently removes most of the most poorly aligned regions.

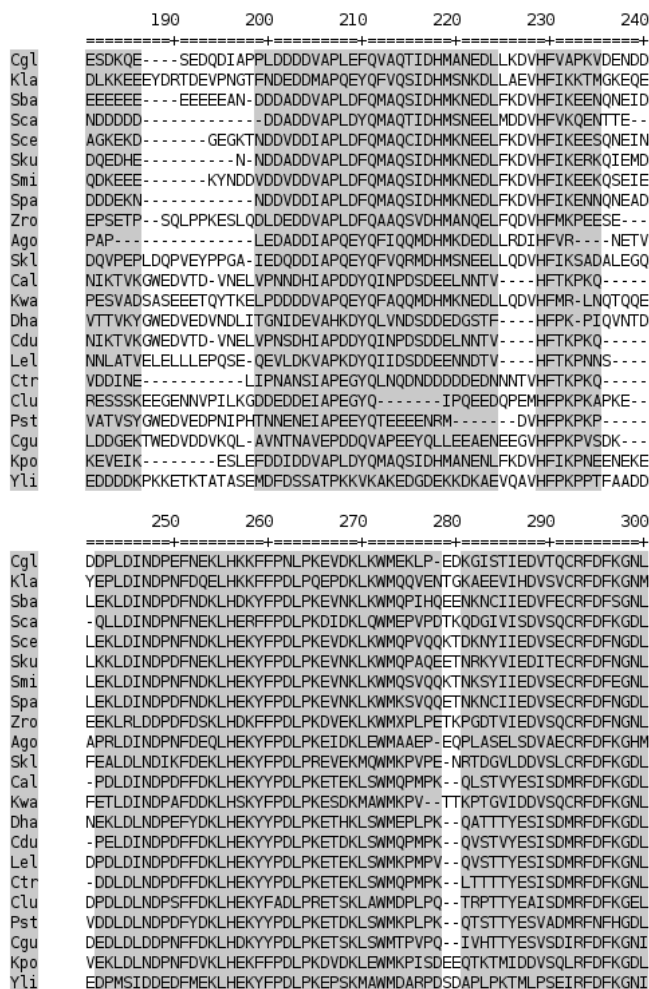


Figure S2. An example of an alignment trimmed with the gappyout method. Conserved (grey) and trimmed (white) columns are indicated. This figure has been generated with *trimAl -htmlout* option.

1.3.4.2 The -strict method.

This method combines a gappyout trimming with a subsequent trimming based on an automatically selected similarity threshold. In order to select the similarity threshold, *trimAl* uses the residue similarity scores distribution from the MSA. This distribution is transformed to a logarithmic scale (see figure S3), and then the residue similarity cut-off is selected as explained below.

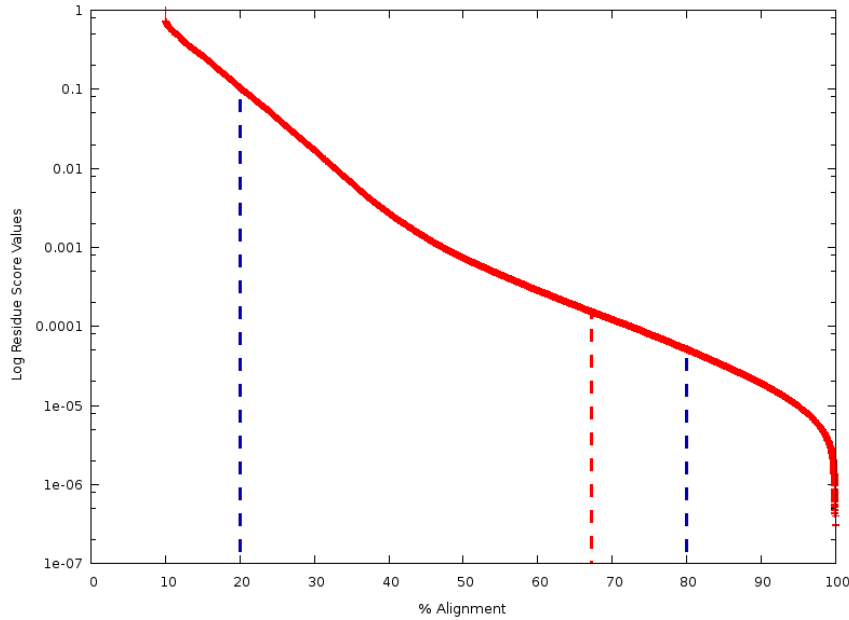


Figure S3. *trimAl*'s internal plot representing similarity score values versus the percentage of the alignment below that value. Vertical blue lines indicate the significant values at 20 and 80 percentiles. The cut-off point is indicated with a red vertical line.

From this similarity distribution, *trimAl* selects the values at the points at percentiles 20 and 80 of the alignment length (vertical blue lines in figure S3), and computes the residue similarity threshold (vertical red line in the figure S3) as follows:

$$P_{20} = \log(\text{SimValue}_{20})$$

$$P_{80} = \log(\text{SimValue}_{80})$$

$$\text{SimThreshold} = \left(\left(\frac{P_{20} - P_{80}}{10} \right) + P_{80} \right)^{10}$$

This is equivalent to setting upper and lower boundaries for the threshold at percentiles 20 and 80, respectively, of the similarity score distribution in that alignment, and then set the similarity threshold so that it is ten times closer to the lower boundary (similarity at P_{80}) than to the upper limit (similarity at P_{20}). This method of setting the similarity threshold has worked best in our benchmarks. The lower and upper boundaries assure that the 20% most conserved columns in the alignment will be conserved, whereas the 20% most dissimilar columns will be discarded. The specific similarity threshold will lie between these boundaries depending on the specific distribution of similarity scores of the alignments: alignments with step similarity score curves and large differences between most similar and most dissimilar columns will set more columns below the threshold, whereas those with more columns with scores similar to the most-conserved fraction will

apply more relaxed cut-offs. However, the specific removal of a column will depend on its context (see below).

Once *trimAl* has calculated the residue similarity cut-off, *trimAl* proceeds as follows: 1) The gappyout method (see above) is applied to mark those columns that would be deleted with that method. 2) The residues that are below the similarity cut-off are also marked. 3) After applying these filters, *trimAl* recovers (unmarks) columns that have not passed the gap and/or similarity thresholds, but where three of the four most immediate neighboring columns (two at each side) have passed them. 4) Finally, in a last step, *trimAl* removes all columns that do not fall within a block of at least five consecutive columns unmarked for deletion.

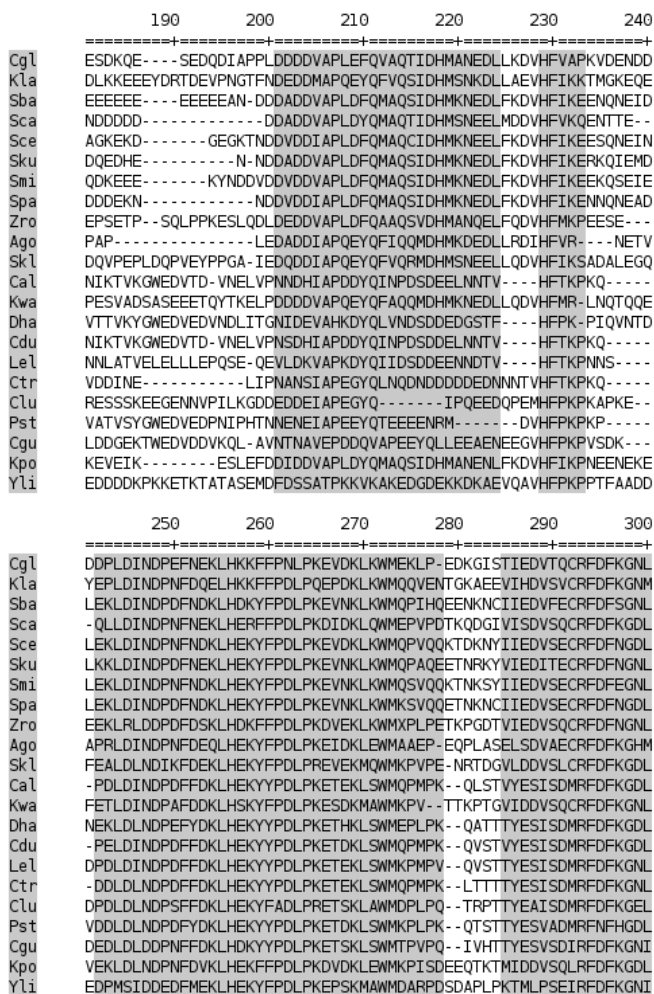


Figure S4. An example of an alignment trimmed with the strict method. We have used the same alignment as in figure S2. Conserved (grey) and trimmed (white) columns are indicated.

1.3.4.3 The -strictplus method (Optimized for Neighbor Joining phylogenetic tree reconstruction).

This method is similar to the -strict method, but they differ in that -strictplus automatically selects the block size for the final step of the algorithm. This block size is

defined as 1% of the alignment size with a minimum value of 3 and a maximum size of 12.

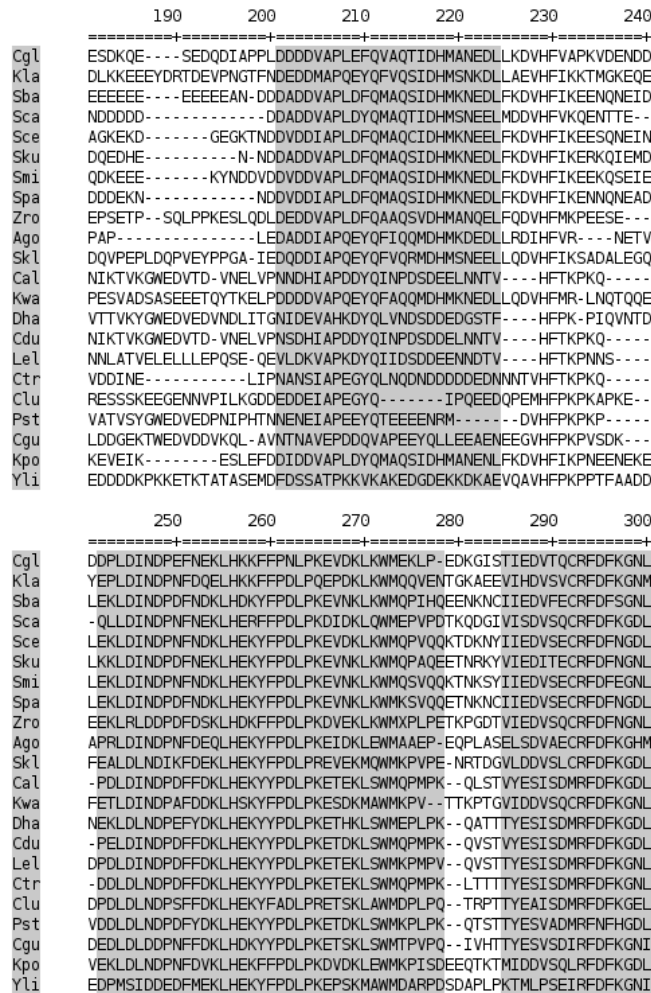


Figure S5. An example of an alignment trimmed with strictplus method. In this case, the block size has automatically been set to 12 because the alignment length is greater than 1200 residues. Again, the same alignment as the previous figures S2 and S4 has been used.

1.3.5 Automated trimming heuristic: automated1. (Optimized for Maximum Likelihood phylogenetic tree reconstruction).

Based on our own benchmarks with simulated alignments (see below and on-line documentation), we have designed a heuristic approach -automated1- in order to select the best automatic method to trim a given alignment. This heuristic is optimized for trimming alignments that will be analyzed by Maximum Likelihood phylogenetic analyses, future releases of *trimAl* may incorporate new heuristics that are optimized for other applications. Using a decision tree (see figure below) this heuristic chooses between gappyout and strict methods (see above). For this, *trimAl* considers the average identity score among all the sequences in the alignment, the average identity score for each most similar pair of sequences in the alignment, as well as the number of sequences in the alignment. We have observed that all these variables were important in deciding which

method would provide the highest improvement on a given alignment.

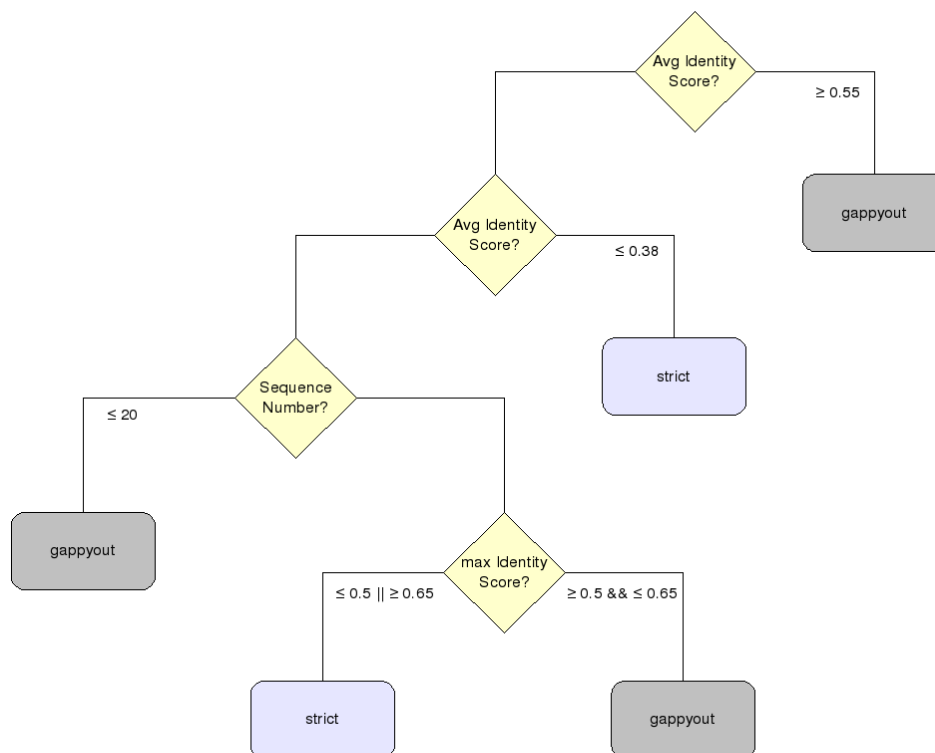


Figure S6. A decision tree for the heuristic method automated1. *trimAl* uses strict (light blue) or gappyout (light grey) methods depending on 1) the average identity score (Avg. identity score) among the sequences in the alignment, 2) the number of sequences in the alignment and 3) the average identity score (max Identity Score) computed from the maximum identity score for each sequence in the alignment. We use light yellow color to highlight the decisions in the tree.

1.3.6 Automated removal of spurious sequences.

trimAl can also remove poorly aligned or incomplete sequences considering the rest of sequences in the MSA. For that purpose, the user has to define two thresholds:

First, the residue overlap threshold (*-resoverlap*) corresponds to the minimum residue overlap score for each residue.

Second, the sequence overlap threshold (*-seqoverlap*) sets up the minimum percentage of the residues for each sequence that should pass the residue overlap threshold in order to maintain the sequence in the new alignment. Sequences that do not pass the sequence overlap threshold will be removed from the alignment. Finally, all columns that only have gaps in the new alignment will also be removed from the final alignment.

```

          52750      52760      52770      52780      52790      52800
=====+=====+=====+=====+=====+=====+
Cgl  LWTKLYHAKYIDYCLYEKLPSSSEYMSLSNMLRASRLAQVYKSSKPLS-TLKIAPTHQVIE
Kla  QWNQVYGVHYIDYCLFEKPTETINMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Sba  QWNQVYGVHYIDYCLFEKPTETINMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Sca  SMESLYGVNYLDYCLFERIHYAPDSRI SNLLRSTRVAQVPKNNKPLNPIHKYFPTHQIIE
Sce  QWNQVYGVRVYIDYCLFEKPTETINMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Sku  -----MTLAELLGRSRIAQVANNHKPLSYTKKFHPTHQIIE
Smi  QWNKQVYGVHYIDYCLFEKPTETIN-----
Spa  QWNQLYGVRVYIDYCLFEKPTETINMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Zro  -----
Ago  LVERYHKAKHLQYCVFKNSAAELASNLSSILRNSRIAQVPKPKGPLFSPKQYVPTHQLIE
Sk1  IWNTVYGVFTEYCLFERKREVV-SNLSHLLRNSRIAHPVPSKPLNSAPRFHPTHQIIE
Ca1  -WLRPYDVGMEFCLLKQDIEDTSELFNLVKNSRLAQVAKPLSNNIRGNSKTPTHQVIF
Kwa  YWTRVYGVLEYCIKREMEEV-SNLSHLLRNSRLAQVPSKPLTSAPKYAPTHQIIE
Dha  SLIP-YNVVMDYCLVERKNI V-QELYNFLKNSKLAQVATPLHKTLRGDNVPSHQIYY
Cdu  -CLRPYDVSMEFCLLKQDIDTSELFNLVKNSRLAQVAKPLSNNIRGNSKTPTHQIIF
Le1  KWIP-YNVVYMQFCILKNRVEKQHRELFDLVKNSKLAQVSTPRSKLIRFSRTKPTHQIYY
Ctr  ----FRVERMDFCVLKKVVDV--NELFNLVKNSRLSQAQPLSKNMRKSNIPTHQVIF
Clu  TVVVPN-SVEYMDFCVLRKK----QEFYKLFKNSKLAQVATPLSKKLRGNTSAPTHQIYY
Pst  LIVR-YNVVYMEFCVLRK----KNQELFNLVNSKLAQVATPLKNIRGKSSVPTHQIIF
Cgu  KWVP-YNVVYMNLCIAEKRSQ--NELYGLLRNSRLAQVATPLSKKLRGESHIPTHQIYY
Kpo  YWNKLYGVNYIDYCVYEKTEIVQDMSVSQLLRKSKIAQVNTKNP-LYVNNKFHPTHQLIT
Yli  GLTK-TSLKHHEYCLFEKV--KNE-----
          52810      52820      52830      52840      52850      52860
=====+=====+=====+=====+=====+=====+
Cgl  TKPALLKFHEWGLKSAIPSKVKTQYLYMYKELDTLERLTDPEPRGGANWNRIRVQEFGLSP
Kla  TKPSSLYRQEWGLKASIPSKIKTRYLVYNDLDTQQRLTTFEPMGQYQWNRI RFQEMGLAP
Sba  TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Sca  TKPSSSHRQEWGLKSSIPSKIKSRYLVFNMDTLERLTTFEPHGASQWNRI RFKEMNLAP
Sce  TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Sku  TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Smi  -----
Spa  TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Zro  -----SSIX-----
Ago  NKASTMHRQEWGMKSSIPSRKSRYLIFDELDTQQRLTSFEAIGQYQWNRI RIQELGVVP
Sk1  TKPSTLHRQEWGLKSSIPSKVQTRYIIFNDLDTLERLTTFEPNAGSQWSRLRFQELGVAP
Ca1  TPKSSALRSYGLKSTLPNKIGSSHISFNIDNRQSMQDVEKNSGFHYKQLMFQELGLCI
Kwa  TRPSTLHRQEWGLKAALPSKTKTKYIVFNDLDTLERLTTFEPSSGGAQWNRI RFQELGIAP
Dha  TPKSSAIRSNFGIKTALPKQIGFSHIVYNDIDNPKNMPDVEYSGPLYNRLKFQETGVAV
Cdu  TPKSSALRSYGLKSTLPNKIGSSHISFNIDNRQSMQDVEKNSGFHYKQLMFQELGLCV
Le1  TPTTSALRSYGMKRTLPAKYGKSHIAFNIDNAKEMPDVEKESSEHYTRLMFQELGKPL
Ctr  TPKSSAARSNYGLKSTLPNKIGSSYISFNIDNRQSMQDVEKNSGFYKQLMFQELGLPV
Clu  TPTASAARSNFGIKTSLPKQIGQSHIVFNIDNFKNMPDVEKHAGPHYTRLKFQESGIVL
Pst  TPKSSAVRSNFGIKTLPKQIGYSHISFNIDNYSMPDVEKNSGMYNRLKFQETGLVV
Cgu  TPKSSAIRSSFGIKTQLPKQVQSHIVFNIDNPKNMPDVEKYSKGYNRLKFQETGMVV
Kpo  TKRTNFNQNDWGLKSPIPSRDNRYLVYNDLDTLERIATYEKNNQQWTRIRFNFELNLT
Yli  -----

```

Figure S7. An example of an alignment trimmed with the option to remove spurious sequences. In this case, we have used these parameter: 1) `-resoverlap 0.75` and 2) `-seqoverlap 75`. Conserved (grey) and trimmed (white) sequences are indicated. Again, the same alignment as the previous figures S2, S4 and S5 has been used.