Supplementary Material for the paper:

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

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

Index of contents

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 *-colnumbering*). *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:

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

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 Si.

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).

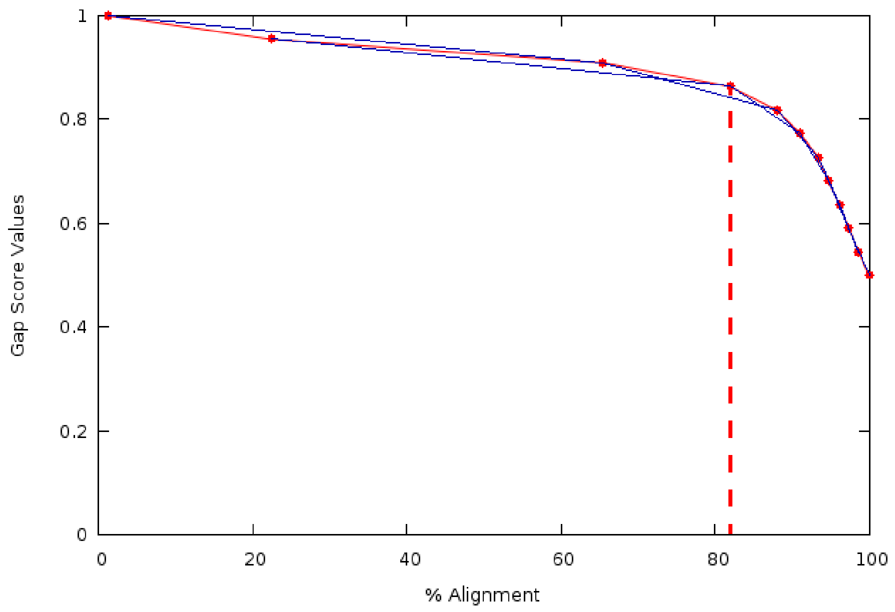
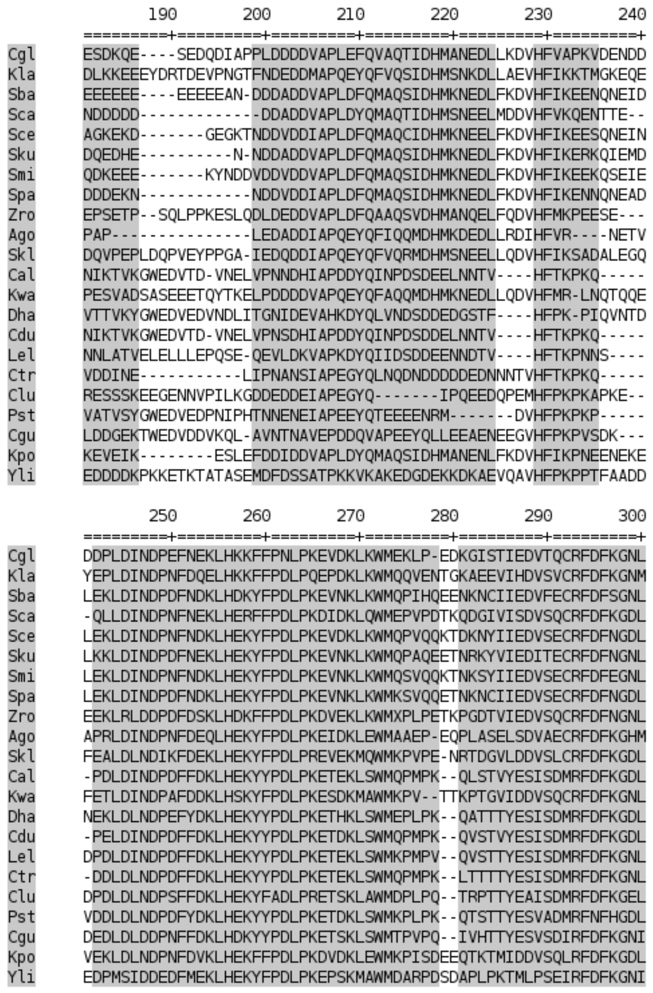


Figure S1. Example of an internal *trimAl* plot showing possible gap score thresholds (y axis) versus percentages of alignment length below that threshold (x axis). Thin blue lines indicate slopes computed by the program. The vertical blue line indicates the cut-off point selected by the gappyout algorithm.

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.



```
                 190       200       210       220       230       240
          =========+=========+=========+=========+=========+=========+
Cgl       ESDKQE----SEDQDIAPPLDDDDVAPLEFQVAQTIDHMANEDLLKDVHFVAPKVDENDD
Kla       DLKKEEEYDRTDEVPNGTFNDEDDMAPQEYQFVQSIDHMSNKDLLAEVHFIKKTMGKEQE
Sba       EEEEEE----EEEEEAN-DDDADDVAPLDFQMAQSIDHMKNEDLFKDVHFIKEENQNEID
Sca       NDDDDD-------------DDADDVAPLDYQMAQTIDHMSNEELMDDVHFVKQENTTE--
Sce       AGKEKD-------GEGKTNDDVDDIAPLDFQMAQCIDHMKNEELFKDVHFIKEESQNEIN
Sku       DQEDHE----------N-NDDADDVAPLDFQMAQSIDHMKNEDLFKDVHFIKERKQIEMD
Smi       QDKEEE-------KYNDDVDDVDDVAPLDFQMAQSIDHMKNEDLFKDVHFIKEEKQSEIE
Spa       DDDEKN------------NDDVDDIAPLDFQMAQSIDHMKNEDLFKDVHFIKENNQNEAD
Zro       EPSETP--SQLPPKESLQDLDEDDVAPLDFQAAQSVDHMANQELFQDVHFMKPEESE---
Ago       PAP---------------LEDADDIAPQEYQFIQQMDHMKDEDLLRDIHFVR----NETV
Skl       DQVPEPLDQPVEYPPGA-IEDQDDIAPQEYQFVQRMDHMSNEELLQDVHFIKSADALEGQ
Cal       NIKTVKGWEDVTD-VNELVPNNDHIAPDDYQINPDSDEELNNTV----HFTKPKQ-----
Kwa       PESVADSASEEETQYTKELPDDDDVAPQEYQFAQQMDHMKNEDLLQDVHFMR-LNQTQQE
Dha       VTTVKYGWEDVEDVNDLITGNIDEVAHKDYQLVNDSDDEDGSTF----HFPK-PIQVNTD
Cdu       NIKTVKGWEDVTD-VNELVPNSDHIAPDDYQINPDSDDELNNTV----HFTKPKQ-----
Lel       NNLATVELELLLEPQSE-QEVLDKVAPKDYQIIDSDDEENNDTV----HFTKPNNS----
Ctr       VDDINE-----------LIPNANSIAPEGYQLNQDNDDDDDEDNNNTVHFTKPKQ-----
Clu       RESSSKEEGENNVPILKGDDEDDEIAPEGYQ-------IPQEEDQPEMHFPKPKAPKE--
Pst       VATVSYGWEDVEDPNIPHTNNENEIAPEEYQTEEEENRM-------DVHFPKPKP-----
Cgu       LDDGEKTWEDVDDVKQL-AVNTNAVEPDDQVAPEEYQLLEEAENEEGVHFPKPVSDK---
Kpo       KEVEIK--------ESLEFDDIDDVAPLDYQMAQSIDHMANENLFKDVHFIKPNEENEKE
Yli       EDDDDKPKKETKTATASEMDFDSSATPKKVKAKEDGDEKKDKAEVQAVHFPKPPTFAADD

                 250       260       270       280       290       300
          =========+=========+=========+=========+=========+=========+
Cgl       DDPLDINDPEFNEKLHKKFFPNLPKEVDKLKWMEKLP-EDKGISTIEDVTQCRFDFKGNL
Kla       YEPLDINDPNFDQELHKKFFPDLPQEPDKLKWMQQVENTGKAEEVIHDVSVCRFDFKGNM
Sba       LEKLDINDPDFNDKLHDKYFPDLPKEVNKLKWMQPIHQEENKNCIIEDVFECRFDFSGNL
Sca       -QLLDINDPNFNEKLHERFFPDLPKDIDKLQWMEPVPDTKQDGIVISDVSQCRFDFKGDL
Sce       LEKLDINDPNFNDKLHEKYFPDLPKEVDKLKWMQPVQQKTDKNYIIEDVSECRFDFNGDL
Sku       LKKLDINDPDFNEKLHEKYFPDLPKEVNKLKWMQPAQEETNRKYVIEDITECRFDFNGNL
Smi       LEKLDINDPNFNDKLHEKYFPDLPKEVNKLKWMQSVQQKTNKSYIIEDVSECRFDFEGNL
Spa       LEKLDINDPDFNDKLHEKYFPDLPKEVNKLKWMKSVQQETNKNCIIEDVSECRFDFNGDL
Zro       EEKLRLDDPDFDSKLHDKFFPDLPKDVEKLKWMXPLPETKPGDTVIEDVSQCRFDFNGNL
Ago       APRLDINDPNFDEQLHEKYFPDLPKEIDKLEWMAAEP-EQPLASELSDVAECRFDFKGHM
Skl       FEALDLNDIKFDEKLHEKYFPDLPREVEKMQWMKPVPE-NRTDGVLDDVSLCRFDFKGDL
Cal       -PDLDINDPDFFDKLHEKYYPDLPKETEKLSWMQPMPK--QLSTVYESISDMRFDFKGDL
Kwa       FETLDINDPAFDDKLHSKYFPDLPKESDKMAWMKPV--TTKPTGVIDDVSQCRFDFKGNL
Dha       NEKLDLNDPEFYDKLHEKYYPDLPKETHKLSWMEPLPK--QATTTYESISDMRFDFKGDL
Cdu       -PELDINDPDFFDKLHEKYYPDLPKETDKLSWMQPMPK--QVSTVYESISDMRFDFKGDL
Lel       DPDLDINDPDFFDKLHEKYYPDLPKETEKLSWMKPMPV--QVSTTYESISDMRFDFKGNL
Ctr       -DDLDLNDPDFFDKLHEKYYPDLPKETEKLSWMQPMPK--LTTTTYESISDMRFDFKGNL
Clu       DPDLDLNDPSFFDKLHEKYFADLPRETSKLAWMDPLPQ--TRPTTYEAISDMRFDFKGEL
Pst       VDDLDLNDPDFYDKLHEKYYPDLPKETDKLSWMKPLPK--QTSTTYESVADMRFNFHGDL
Cgu       DEDLDLDDPNFFDKLHDKYYPDLPKETSKLSWMTPVPQ--IVHTTYESVSDIRFDFKGNI
Kpo       VEKLDLNDPNFDVKLHEKFFPDLPKDVDKLEWMKPISDEEQTKTMIDDVSQLRFDFKGDL
Yli       EDPMSIDDEDFMEKLHEKYFPDLPKEPSKMAWMDARPDSDAPLPKTMLPSEIRFDFKGNI
```

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.

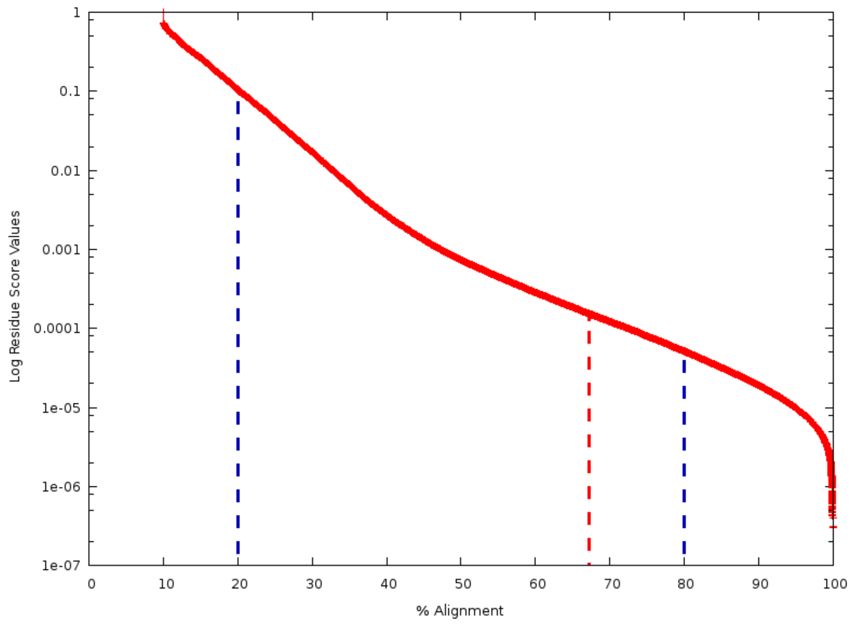


Figure S3. *trimAl*'s internal plot representing similarity score values versus the percentage of the alignment below that value. Vertical blue lines indicate the significant values at 20 and 80 percentiles. The cut-off point is indicated with a red vertical line.

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:

$$P_{20} = \log(SimValue_{20})$$

$$P_{80} = \log(SimValue_{80})$$

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

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.

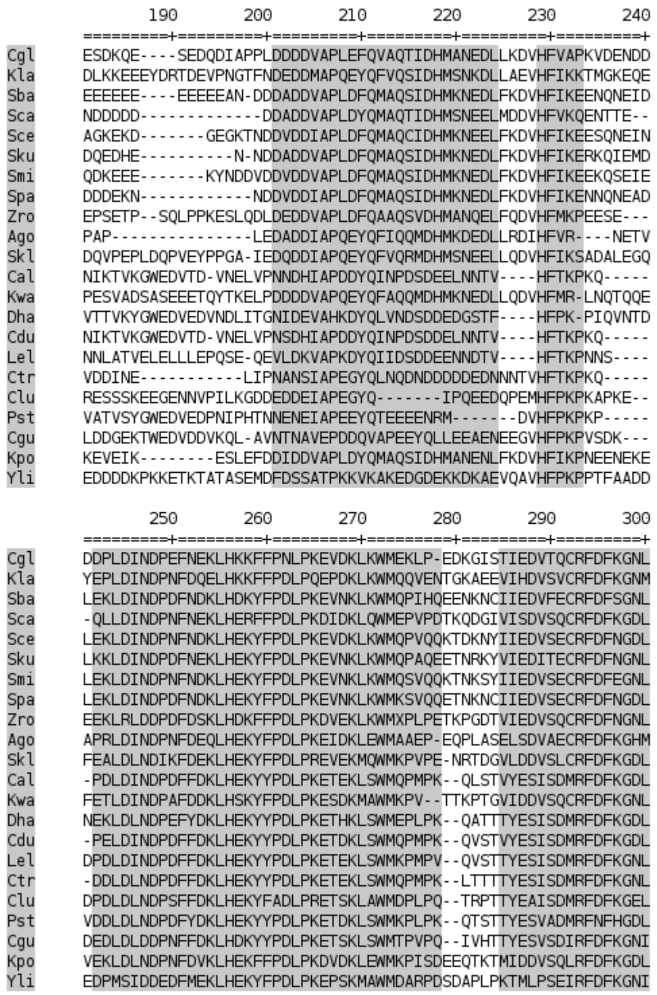
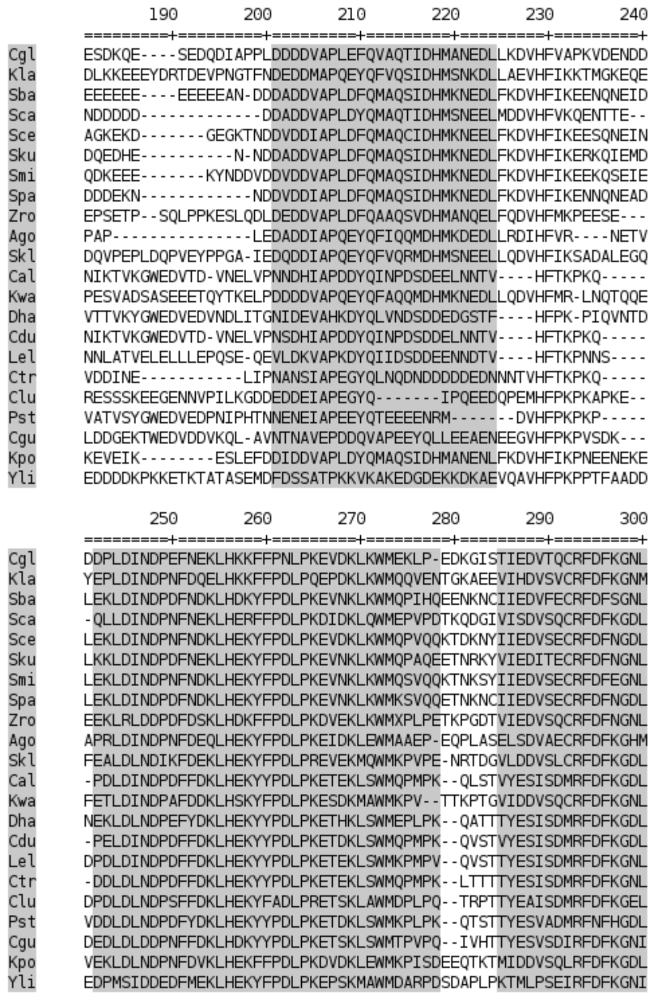


Figure S4. An example of an alignment trimmed with the strict method. We have used the same alignment as in figure S2. Conserved (grey) and trimmed (white) columns are indicated.

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.



```
               190       200       210       220       230       240
        =========+=========+=========+=========+=========+=========+
Cgl     ESDKQE----SEDQDIAPPLDDDDVAPLEFQVAQTIDHMANEDLLKDVHFVAPKVDENDD
Kla     DLKKEEEYDRTDEVPNGTFNDEDDMAPQEYQFVQSIDHMSNKDLLAEVHFIKKTMGKEQE
Sba     EEEEEE----EEEEEAN-DDDADDVAPLDFQMAQSIDHMKNEDLFKDVHFIKEENQNEID
Sca     NDDDDD-------------DDADDVAPLDYQMAQTIDHMSNEELMDDVHFVKQENTTE--
Sce     AGKEKD-------GEGKTNDDVDDIAPLDFQMAQCIDHMKNEELFKDVHFIKEESQNEIN
Sku     DQEDHE----------N-NDDADDVAPLDFQMAQSIDHMKNEDLFKDVHFIKERKQIEMD
Smi     QDKEEE-------KYNDDVDDVDDVAPLDFQMAQSIDHMKNEDLFKDVHFIKEEKQSEIE
Spa     DDDEKN------------NDDVDDIAPLDFQMAQSIDHMKNEDLFKDVHFIKENNQNEAD
Zro     EPSETP--SQLPPKESLQDLDEDDVAPLDFQAAQSVDHMANQELFQDVHFMKPEESE---
Ago     PAP---------------LEDADDIAPQEYQFIQQMDHMKDEDLLRDIHFVR----NETV
Skl     DQVPEPLDQPVEYPPGA-IEDQDDIAPQEYQFVQRMDHMSNEELLQDVHFIKSADALEGQ
Cal     NIKTVKGWEDVTD-VNELVPNNDHIAPDDYQINPDSDEELNNTV----HFTKPKQ-----
Kwa     PESVADSASEEETQYTKELPDDDDVAPQEYQFAQQMDHMKNEDLLQDVHFMR-LNQTQQE
Dha     VTTVKYGWEDVEDVNDLITGNIDEVAHKDYQLVNDSDDEDGSTF----HFPK-PIQVNTD
Cdu     NIKTVKGWEDVTD-VNELVPNSDHIAPDDYQINPDSDDELNNTV----HFTKPKQ-----
Lel     NNLATVELELLLEPQSE-QEVLDKVAPKDYQIIDSDDEENNDTV----HFTKPNNS----
Ctr     VDDINE-----------LIPNANSIAPEGYQLNQDNDDDDDEDNNNTVHFTKPKQ-----
Clu     RESSSKEEGENNVPILKGDDEDDEIAPEGYQ-------IPQEEDQPEMHFPKPKAPKE--
Pst     VATVSYGWEDVEDPNIPHTNNENEIAPEEYQTEEEENRM-------DVHFPKPKP-----
Cgu     LDDGEKTWEDVDDVKQL-AVNTNAVEPDDQVAPEEYQLLEEAENEEGVHFPKPVSDK---
Kpo     KEVEIK--------ESLEFDDIDDVAPLDYQMAQSIDHMANENLFKDVHFIKPNEENEKE
Yli     EDDDDKPKKETKTATASEMDFDSSATPKKVKAKEDGDEKKDKAEVQAVHFPKPPTFAADD

               250       260       270       280       290       300
        =========+=========+=========+=========+=========+=========+
Cgl     DDPLDINDPEFNEKLHKKFFPNLPKEVDKLKWMEKLP-EDKGISTIEDVTQCRFDFKGNL
Kla     YEPLDINDPNFDQELHKKFFPDLPQEPDKLKWMQQVENTGKAEEVIHDVSVCRFDFKGNM
Sba     LEKLDINDPDFNDKLHDKYFPDLPKEVNKLKWMQPIHQEENKNCIIEDVFECRFDFSGNL
Sca     -QLLDINDPNFNEKLHERFFPDLPKDIDKLQWMEPVPDTKQDGIVISDVSQCRFDFKGDL
Sce     LEKLDINDPNFNDKLHEKYFPDLPKEVDKLKWMQPVQQKTDKNYIIEDVSECRFDFNGDL
Sku     LKKLDINDPDFNEKLHEKYFPDLPKEVNKLKWMQPAQEETNRKYVIEDITECRFDFNGNL
Smi     LEKLDINDPNFNDKLHEKYFPDLPKEVNKLKWMQSVQQKTNKSYIIEDVSECRFDFEGNL
Spa     LEKLDINDPDFNDKLHEKYFPDLPKEVNKLKWMKSVQQETNKNCIIEDVSECRFDFNGDL
Zro     EEKLRLDDPDFDSKLHDKFFPDLPKDVEKLKWMXPLPETKPGDTVIEDVSQCRFDFNGNL
Ago     APRLDINDPNFDEQLHEKYFPDLPKEIDKLEWMAAEP-EQPLASELSDVAECRFDFKGHM
Skl     FEALDLNDIKFDEKLHEKYFPDLPREVEKMQWMKPVPE-NRTDGVLDDVSLCRFDFKGDL
Cal     -PDLDINDPDFFDKLHEKYYPDLPKETEKLSWMQPMPK--QLSTVYESISDMRFDFKGDL
Kwa     FETLDINDPAFDDKLHSKYFPDLPKESDKMAWMKPV--TTKPTGVIDDVSQCRFDFKGNL
Dha     NEKLDLNDPEFYDKLHEKYYPDLPKETHKLSWMEPLPK--QATTTYESISDMRFDFKGDL
Cdu     -PELDINDPDFFDKLHEKYYPDLPKETDKLSWMQPMPK--QVSTVYESISDMRFDFKGDL
Lel     DPDLDINDPDFFDKLHEKYYPDLPKETEKLSWMKPMPV--QVSTTYESISDMRFDFKGNL
Ctr     -DDLDLNDPDFFDKLHEKYYPDLPKETEKLSWMQPMPK--LTTTTYESISDMRFDFKGNL
Clu     DPDLDLNDPSFFDKLHEKYFADLPRETSKLAWMDPLPQ--TRPTTYEAISDMRFDFKGEL
Pst     VDDLDLNDPDFYDKLHEKYYPDLPKETDKLSWMKPLPK--QTSTTYESVADMRFNFHGDL
Cgu     DEDLDLDDPNFFDKLHDKYYPDLPKETSKLSWMTPVPQ--IVHTTYESVSDIRFDFKGNI
Kpo     VEKLDLNDPNFDVKLHEKFFPDLPKDVDKLEWMKPISDEEQTKTMIDDVSQLRFDFKGDL
Yli     EDPMSIDDEDFMEKLHEKYFPDLPKEPSKMAWMDARPDSDAPLPKTMLPSEIRFDFKGNI
```

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.

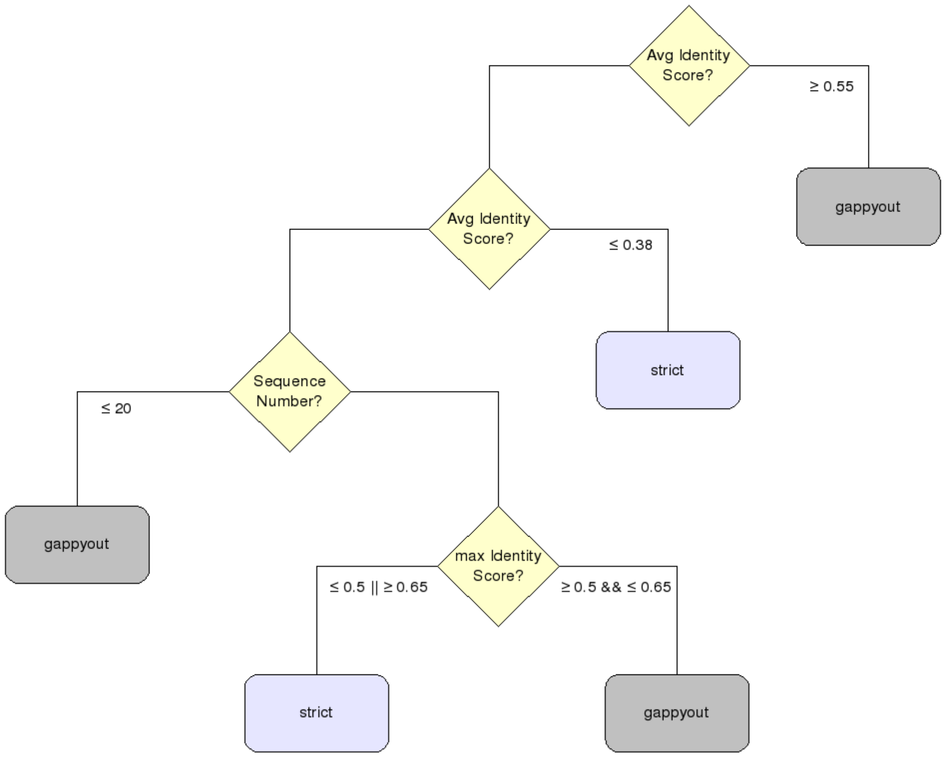


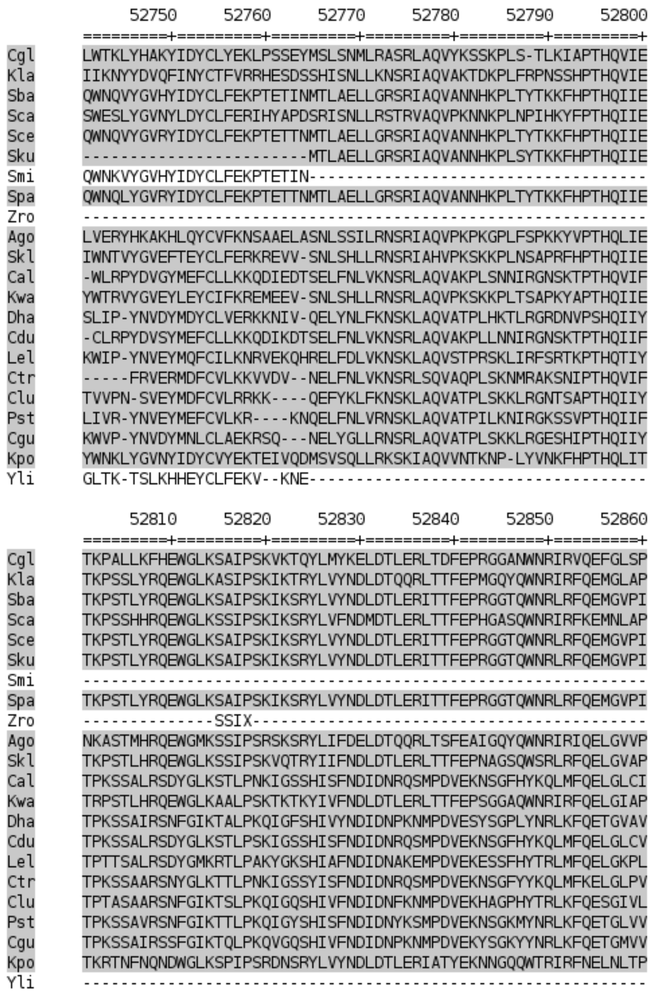
Figure S6. A decision tree for the heuristic method automated1. *trimAl* uses strict (light blue) or gappyout (light grey) methods depending on 1) the average identity score (Avg. identity score) among the sequences in the alignment, 2) the number of sequences in the alignment and 3) the average identity 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       LWTKLYHAKYIDYCLYEKLPSSEYMSLSNMLRASRLAQVYKSSKPLS-TLKIAPTHQVIE
Kla       IIKNYYDVQFINYCTFVRRHESDSSHISNLLKNSRIAQVAKTDKPLFRPNSSHPTHQVIE
Sba       QWNQVYGVHYIDYCLFEKPTETINMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Sca       SWESLYGVNYLDYCLFERIHYAPDSRISNLLRSTRVAQVPKNNKPLNPIHKYFPTHQIIE
Sce       QWNQVYGVRYIDYCLFEKPTETTNMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Sku       ------------------------MTLAELLGRSRIAQVANNHKPLSYTKKFHPTHQIIE
Smi       QWNKVYGVHYIDYCLFEKPTETIN------------------------------------
Spa       QWNQLYGVRYIDYCLFEKPTETTNMTLAELLGRSRIAQVANNHKPLTYTKKFHPTHQIIE
Zro       ------------------------------------------------------------
Ago       LVERYHKAKHLQYCVFKNSAAELASNLSSILRNSRIAQVPKPKGPLFSPKKYVPTHQLIE
Skl       IWNTVYGVEFTEYCLFERKREVV-SNLSHLLRNSRIAHVPKSKKPLNSAPRFHPTHQIIE
Cal       -WLRPYDVGYMEFCLLKKQDIEDTSELFNLVKNSRLAQVAKPLSNNIRGNSKTPTHQVIF
Kwa       YWTRVYGVEYLEYCIFKREMEEV-SNLSHLLRNSRLAQVPKSKKPLTSAPKYAPTHQIIE
Dha       SLIP-YNVDYMDYCLVERKKNIV-QELYNLFKNSKLAQVATPLHKTLRGRDNVPSHQIIY
Cdu       -CLRPYDVSYMEFCLLKKQDIKDTSELFNLVKNSRLAQVAKPLLNNIRGNSKTPTHQIIF
Lel       KWIP-YNVEYMQFCILKNRVEKQHRELFDLVKNSKLAQVSTPRSKLIRFSRTKPTHQTIY
Ctr       -----FRVERMDFCVLKKVVDV--NELFNLVKNSRLSQVAQPLSKNMRAKSNIPTHQVIF
Clu       TVVPN-SVEYMDFCVLRRKK----QEFYKLFKNSKLAQVATPLSKKLRGNTSAPTHQIIY
Pst       LIVR-YNVEYMEFCVLKR----KNQELFNLVRNSKLAQVATPILKNIRGKSSVPTHQIIF
Cgu       KWVP-YNVDYMNLCLAEKRSQ---NELYGLLRNSRLAQVATPLSKKLRGESHIPTHQIIY
Kpo       YWNKLYGVNYIDYCVYEKTEIVQDMSVSQLLRKSKIAQVVNTKNP-LYVNKFHPTHQLIT
Yli       GLTK-TSLKHHEYCLFEKV--KNE------------------------------------

               52810     52820     52830     52840     52850     52860
          =========+=========+=========+=========+=========+=========+
Cgl       TKPALLKFHEWGLKSAIPSKVKTQYLMYKELDTLERLTDFEPRGGANWNRIRVQEFGLSP
Kla       TKPSSLYRQEWGLKASIPSKIKTRYLVYNDLDTQQRLTTFEPMGQYQWNRIRFQEMGLAP
Sba       TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Sca       TKPSSHHRQEWGLKSSIPSKIKSRYLVFNDMDTLERLTTFEPHGASQWNRIRFKEMNLAP
Sce       TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Sku       TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Smi       ------------------------------------------------------------
Spa       TKPSTLYRQEWGLKSAIPSKIKSRYLVYNDLDTLERITTFEPRGGTQWNRLRFQEMGVPI
Zro       --------------SSIX------------------------------------------
Ago       NKASTMHRQEWGMKSSIPSRSKSRYLIFDELDTQQRLTSFEAIGQYQWNRIRIQELGVVP
Skl       TKPSTLHRQEWGLKSSIPSKVQTRYIIFNDLDTLERLTTFEPNAGSQWSRLRFQELGVAP
Cal       TPKSSALRSDYGLKSTLPNKIGSSHISFNDIDNRQSMPDVEKNSGFHYKQLMFQELGLCI
Kwa       TRPSTLHRQEWGLKAALPSKTKTKYIVFNDLDTLERLTTFEPSGGAQWNRIRFQELGIAP
Dha       TPKSSAIRSNFGIKTALPKQIGFSHIVYNDIDNPKNMPDVESYSGPLYNRLKFQETGVAV
Cdu       TPKSSALRSDYGLKSTLPSKIGSSHISFNDIDNRQSMPDVEKNSGFHYKQLMFQELGLCV
Lel       TPTTSALRSDYGMKRTLPAKYGKSHIAFNDIDNAKEMPDVEKESSFHYTRLMFQELGKPL
Ctr       TPKSSAARSNYGLKTTLPNKIGSSYISFNDIDNRQSMPDVEKNSGFYYKQLMFKELGLPV
Clu       TPTASAARSNFGIKTSLPKQIGQSHIVFNDIDNFKNMPDVEKHAGPHYTRLKFQESGIVL
Pst       TPKSSAVRSNFGIKTTLPKQIGYSHISFNDIDNYKSMPDVEKNSGKMYNRLKFQETGLVV
Cgu       TPKSSAIRSSFGIKTQLPKQVGQSHIVFNDIDNPKNMPDVEKYSGKYYNRLKFQETGMVV
Kpo       TKRTNFNQNDWGLKSPIPSRDNSRYLVYNDLDTLERIATYEKNNGQQWTRIRFNELNLTP
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.

2. Benchmark analysis

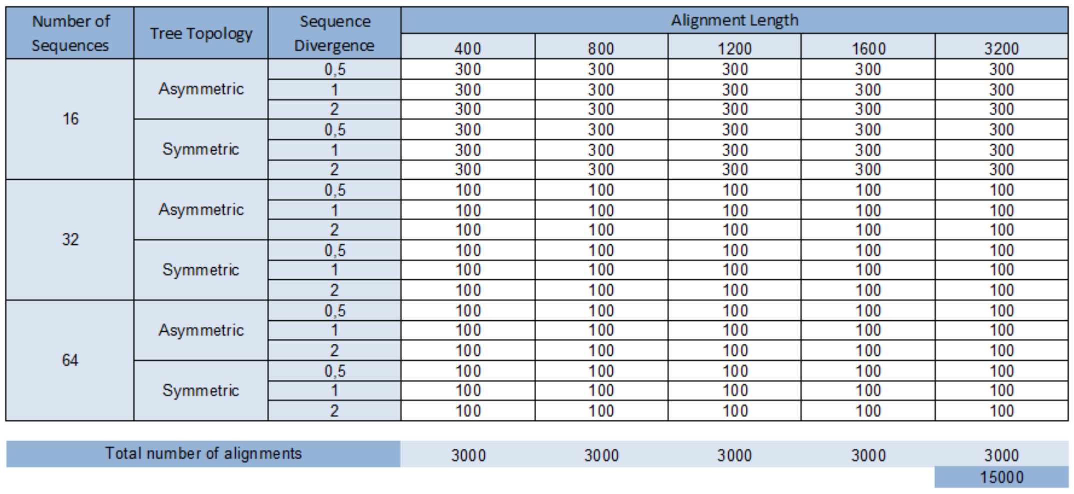
2.1 Construction of the benchmark dataset

In order to test the general applicability of *trimAl*, as well as to find an empirical base to set the heuristics for the automatic selection of parameters we performed a benchmark analysis.

For this purpose we used three different sets to execute our benchmark. One of these sets has been used previously [Talavera G, Castresana J., 2007] to test the improvement in phylogenetic performance after an alignment trimming phase. This set comprises evolutionary simulations of protein sequences of various lengths (400 to 3200 positions), performed with ROSE [Stoye J et al, 1998] along phylogenetic trees with 16 tips. These trees have three different topologies varying in their level of symmetry, and whose branch lengths were multiplied by 0.5, 1 and 2, respectively, totaling six different phylogenetic trees.

The other two sets of alignments were generated by us to expand the dataset to the case of 32 and 64 tree tips, which we consider to be more realistic in phylogenomic analyses. In order to generate these additional sets of alignments, we first took the reference trees from the previous study, one tree for topology. Using ETE [ete.cgenomics.org] we generated twelve new reference trees with the same level of asymmetry as those in the Talavera et. al. study, six of them with 32 tips and the other six with 64 tips. We multiplied the lengths of their branches to obtain the same three levels of divergence (0.5, 1 and 2.0) as the previous study.

This set of reference trees were used to generate the corresponding sets of alignments as indicated in Talavera et al (2007). For this we used the program ROSE v1.3 [Stoye J et al, 1998] using the same seed protein and parameters described in [Talavera G, Castresana J., 2007] to generate their benchmark sets. The simulations included insertions and deletions with a probability of 0.03. The other parameters for the simulation were the ones described in [Talavera G, Castresana J., 2007]. We also used the same strategy to infer the patterns of rate heterogeneity of the seed proteins. Finally, the sets generated by ROSE contain, similarly to the set of 16 sequences, simulated protein sequences of various lengths (400 to 3200 positions) and different topologies.



Number of Sequences	Tree Topology	Sequence Divergence	Alignment Length				
			400	800	1200	1600	3200
16	Asymmetric	0,5	300	300	300	300	300
		1	300	300	300	300	300
		2	300	300	300	300	300
	Symmetric	0,5	300	300	300	300	300
		1	300	300	300	300	300
		2	300	300	300	300	300
32	Asymmetric	0,5	100	100	100	100	100
		1	100	100	100	100	100
		2	100	100	100	100	100
	Symmetric	0,5	100	100	100	100	100
		1	100	100	100	100	100
		2	100	100	100	100	100
64	Asymmetric	0,5	100	100	100	100	100
		1	100	100	100	100	100
		2	100	100	100	100	100
	Symmetric	0,5	100	100	100	100	100
		1	100	100	100	100	100
		2	100	100	100	100	100
Total number of alignments			3000	3000	3000	3000	3000
							15000

Table 1. A summary of the total number of alignments used in our benchmark sorted in different groups depending on the number of sequences, kind of topology, tree divergence and average length of the sequences before being aligned. The last row represents the total number of alignment for each sequences length.

2.2 Phylogenetic analyses.

To measure the improvement in phylogenetic reconstruction after running *trimAl* in the alignments we applied a standard phylogenetic analysis pipeline to each simulated sequence set. This included multiple sequence alignment with MUSCLE v3.7 [Edgar R.C., 2004] and Neighbor Joining or Maximum Likelihood phylogenetic reconstruction using PhyML v2.4 [Guindon S. and Gascuel O., 2003]. For the reconstruction of phylogenetic trees we used the following parameters: 1) datatype: 1, 2) format: i, 3) number of data sets: 1, 4) number of bootstrap data sets to generate: 0, 5) substitution model name: JTT, 6) proportion of invariable sites: e, 7) number of relative substitution rate categories: 4, 8) gamma distribution parameter: e, 9) starting tree: BIONJ 10) optimize tree topology: y (ML) or n (NJ) and 11) optimize branch lengths and rate parameters: y (ML) or n (NJ).

Before the phylogenetic analyses, multiple sequence alignments were trimmed using the Gblocks v0.91b [Castresana J., 2000] with default parameters and *trimAl*'s gappyout, strict, strictplus automated methods and *trimAl*'s automated1 heuristic method. For the summary, we chose the *trimAl*'s strictplus method for neighbor joining tree reconstruction and *trimAl*'s automated1 method for maximum likelihood tree reconstruction method, as they were the optimal choices for these conditions.

The accuracy of the resulting trees was measured by comparing them with the original trees used to generate the sequence sets, and measuring the Robinson Foulds distance using Ktreedist v1.0 program [Soria-Carrasco V. et al, 2007]. The trees produced by the complete, untrimmed, alignment were also compared with the original trees.

In total 180 different conditions were tested, corresponding to different scenarios of tree topology, reconstruction method, alignment length, number of sequences and sequence divergence. Each condition included from 100 to 300 alignments and was tested with Gblocks default parameters and the strictplus (NJ) and automated1 (ML) methods in *trimAl*. Additional benchmark analyses, based on simulated sequences and on real datasets, done in previous versions of *trimAl* guided us in choosing the different options for the parameters. For the sake of simplicity we show here a summary of the last benchmark analyses performed with *trimAl* v1.2. Results from other benchmarks can be seen through the *trimAl* website.

2.3 Results

A summary of the results from this benchmark analyses is represented in the appendix section. In all scenarios considered, the use of *trimAl* either improves the phylogenetic signal or maintains a similar signal to noise ratio relative to the original alignment. In contrast, the use of GBlocks may significantly reduce the phylogenetic signal to noise ratio in the trimmed alignments. This is especially true for the case of Maximimum Likelihood (ML), where untrimmed alignments produce reasonable results but the use of Gblocks may significantly reduce the phylogenetic signal.

In most scenarios (90%) the use of *trimAl* produces better results than GBlocks. In a reduced number of cases (10%), however, Gblocks outperforms *trimAl*, namely, when divergence is high (≥ 2.0), NJ is used for phylogenetic reconstruction, the number of sequences is high (≥ 32) and the alignment length is high (≥ 800 positions). For proteins of average size, e.g. roughly 400 residues in the human genome, *trimAl* performed better in all cases.

The greatest improvements of phylogenetic signal by trimming are achieved when using Neighbor Joining (NJ) and highly divergent trees. When using ML for tree reconstruction, the improvement by trimming is only significant with a combination of moderate to high divergence (≥ 1.0) and large number of sequences. Altogether these results indicate that trimming with *trimAl* strict method would be advisable when NJ is going to be used in the phylogenetic reconstruction, whereas *trimAl* automated1 method would be more indicated when ML is chosen as the tree reconstruction method.

We also compared the resulting using the Ktree score, a measure that takes into account differences in branch lengths [Soria-Carrasco et. al 2007]. The results show, as expected, larger differences in branch lengths when using NJ method for tree reconstruction. With few exceptions, alignments trimmed with *trimAl* present more similarities to the reference tree as compared to the tree reconstructed with the original alignment. The use of Gblocks can produce better results than *trimAl* in terms of Ktree scores when NJ is used on asymmetric trees. In the rest of the conditions *trimAl* produce better results or is comparable to Gblocks. Of important note is that the use of Gblocks on symmetric trees or with ML reconstruction will often result in significantly worse Ktree scores as compared to the untrimmed alignment.

3. Literature cited

Castresana, J. (2000), Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis, Mol Biol Evol, 17 (4), 540-52.

Edgar R.C. (2004), MUSCLE: a multiple sequence alignment method with reduced time and space complexity, BMC Bioinformatics, 5:113.

Guindon S., Gascuel O. (2003), A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood, Syst Biol, 52 (5), 696-704.

Huerta-Cepas, J., et al. (2007), The human phylome, Genome Biol, 8 (6), R109.

Soria-Carrasco V. et al, (2007), The K tree score: quantification of differences in the relative branch length and topology of phylogenetic trees, Bioinformatics, 23 (21), 2954-2956

Stoye, J., Evers, D., Meyer, F. (1998), Rose: generating sequence families, Bioinformatics, 14 (2), 157-63.

Talavera, G., Castresana, J. (2007), Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments, Syst Biol, 56 (4), 564-77.

4. Appendix

The first set of 4 figures (figures S8 to S11) describes the comparisons in terms of topology. Each figure corresponds to a possible combination between the phylogenetic tree reconstruction method (Neighbor Joining or Maximum Likelihood) and the tree topology (Asymmetric or Symmetric). Panels within a figure represent the nine different combinations of the number of sequences in an alignment (16, 32 and 64) and the evolutionary divergence of the seed tree used to generate those alignments (0.5, 1 or 2).

In each panel, x-axis represents the average length of the sequences in the multiple sequence alignment, whereas the y-axis represents the Robinson Foulds distance. This distance measures the topological difference between two given tree, therefore, lower values indicate a better performance of the alignment when reconstructing the tree. Finally, there are three lines that represents the performance of each method: untrimmed alignments (green), Gblocks trimming (blue) and *trimAl* trimming (brown).

The second set of 4 figures (figures S12 to S15) describes the results in terms of Ktree score, which takes into account differences in branch lengths. Figures are organized as in the first set. However, y-axis corresponds in this case to Ktree scores.

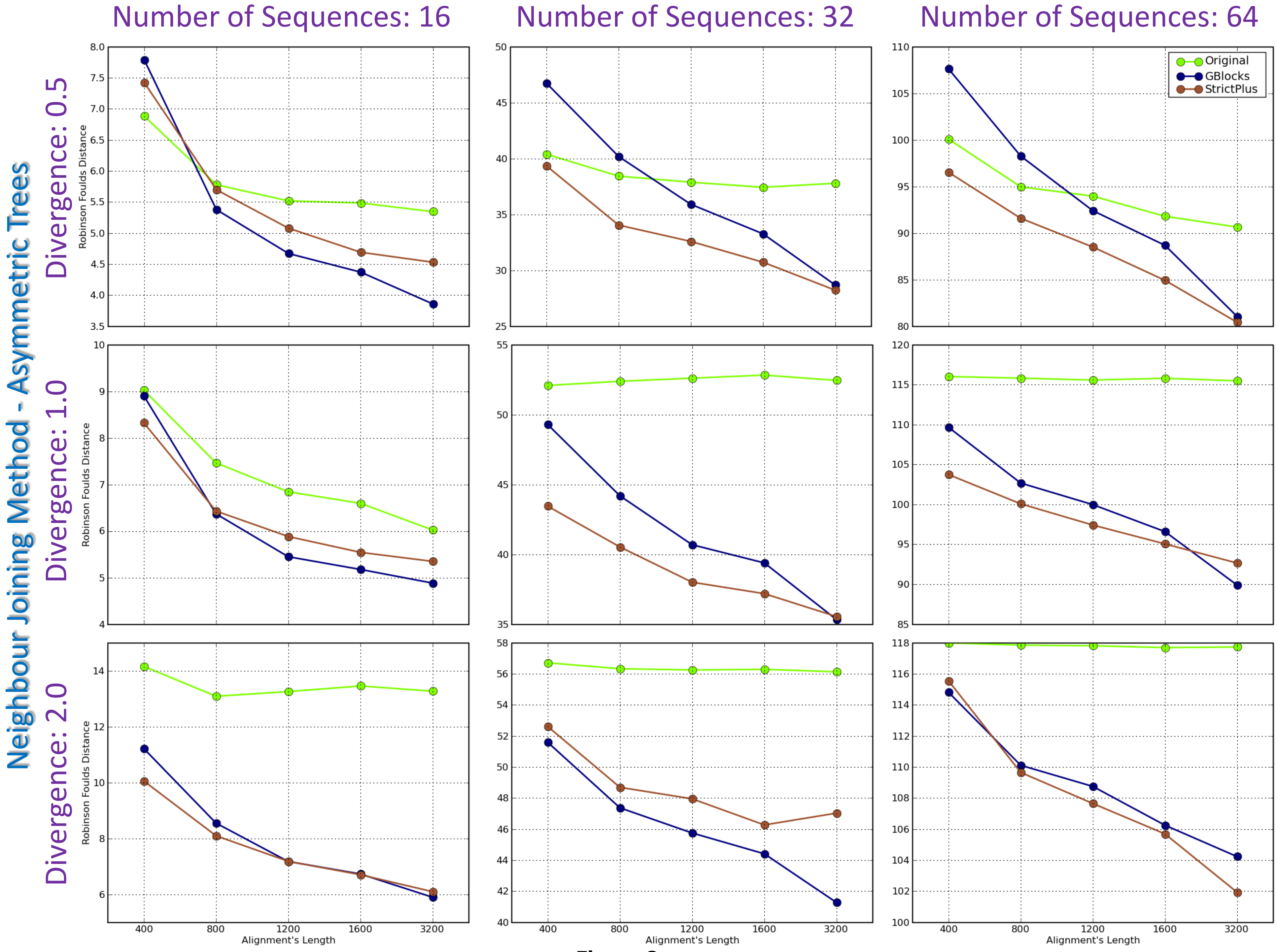
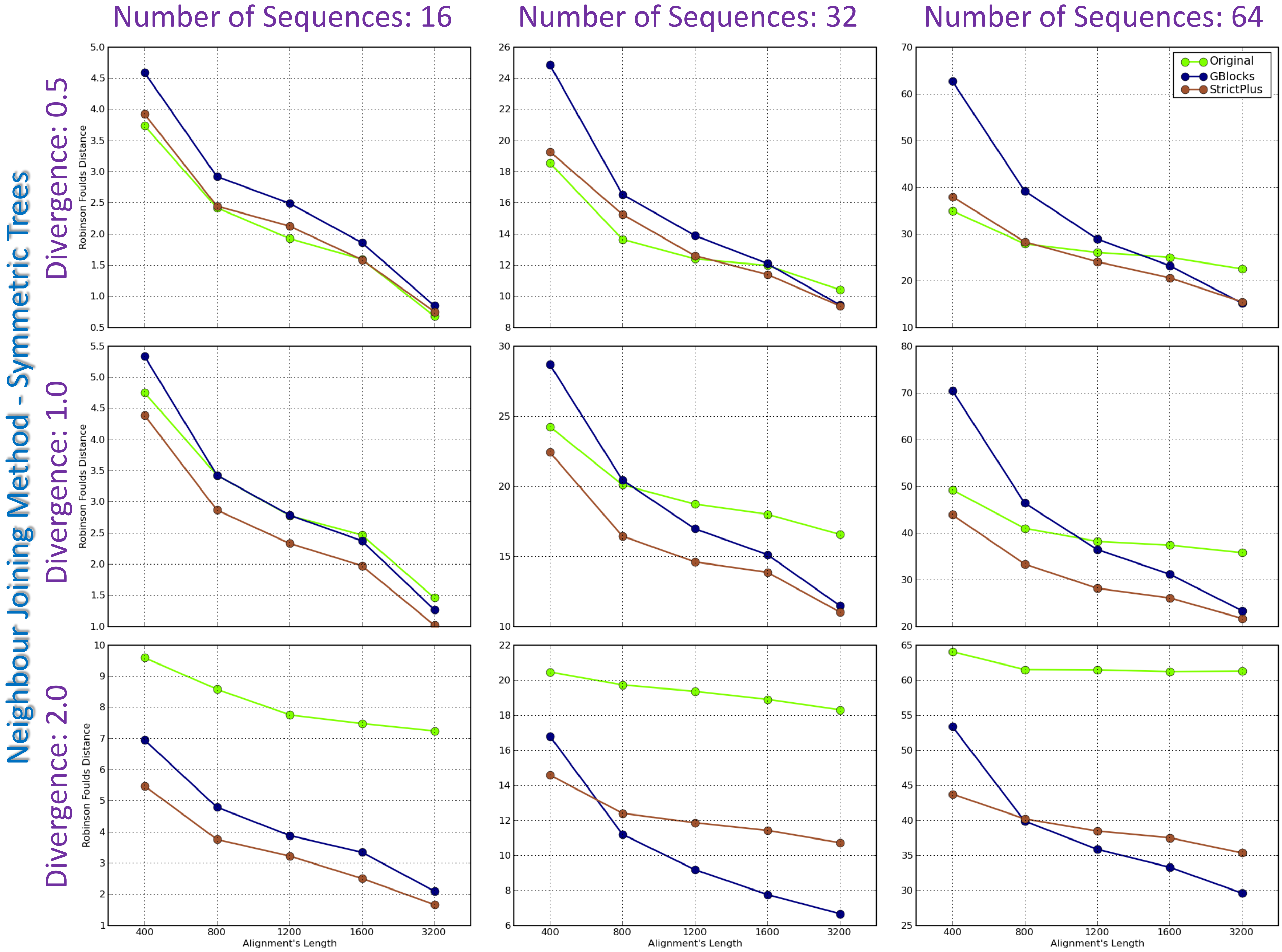

Neighbour Joining Method - Asymmetric Trees
Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
Robinson Foulds Distance
Alignment's Length
Original
GBlocks
StrictPlus

Figure 8



Neighbour Joining Method - Symmetric Trees
Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
Robinson Foulds Distance
Alignment's Length
Original
GBlocks
StrictPlus

Figure 9

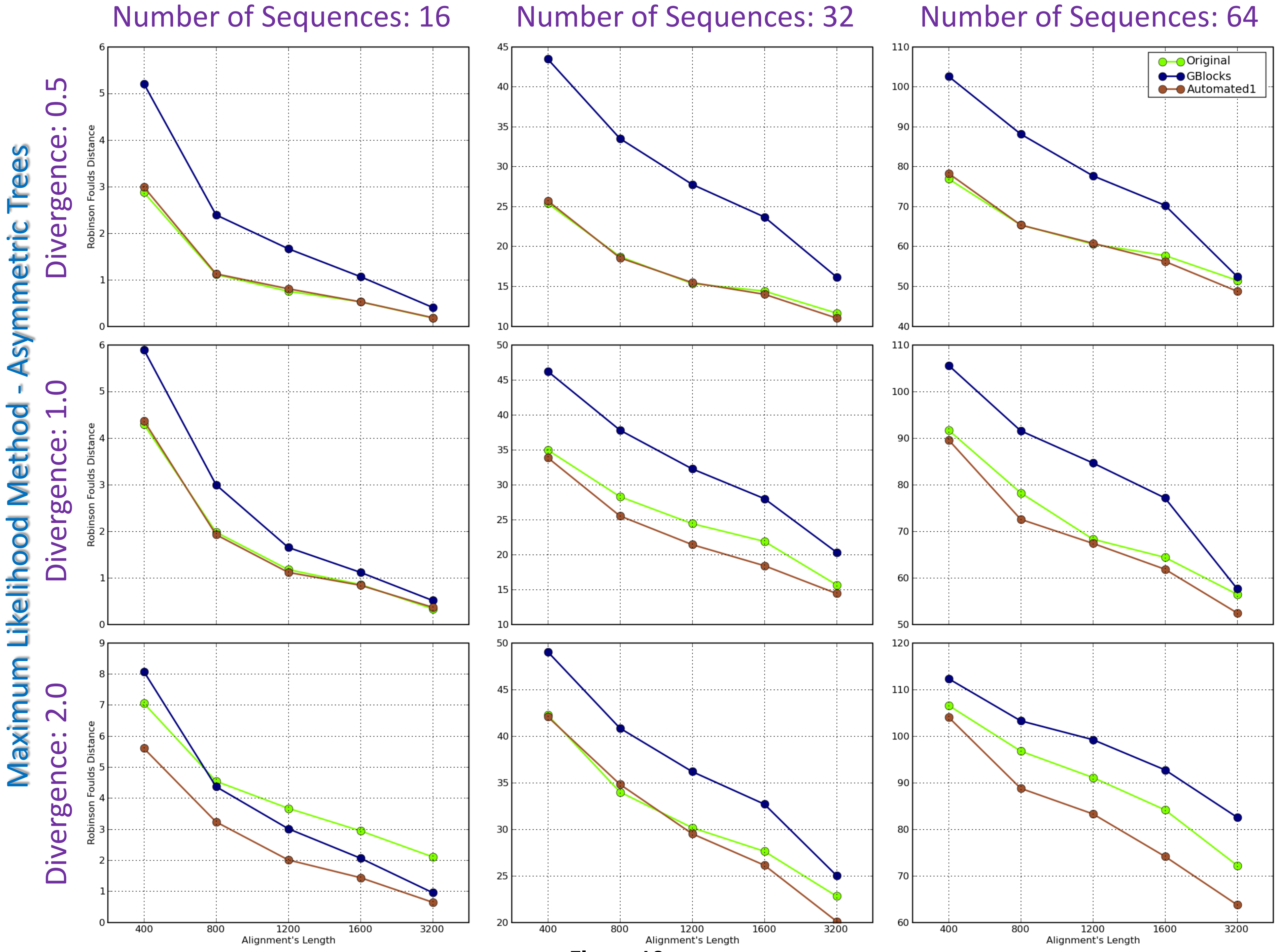

Maximum Likelihood Method - Asymmetric Trees
Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
Robinson Foulds Distance
Alignment's Length
Original
GBlocks
Automated1

Figure 10

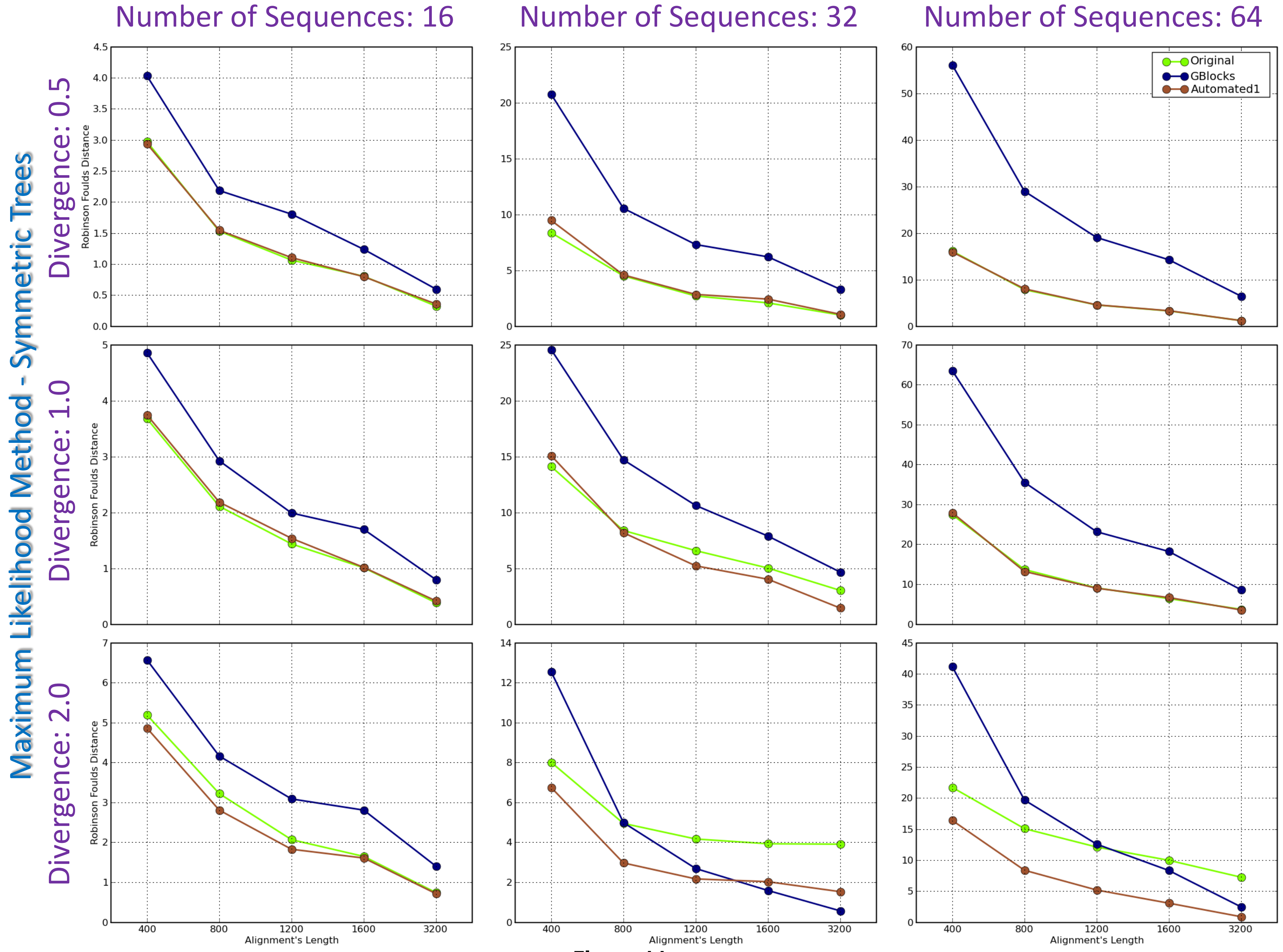
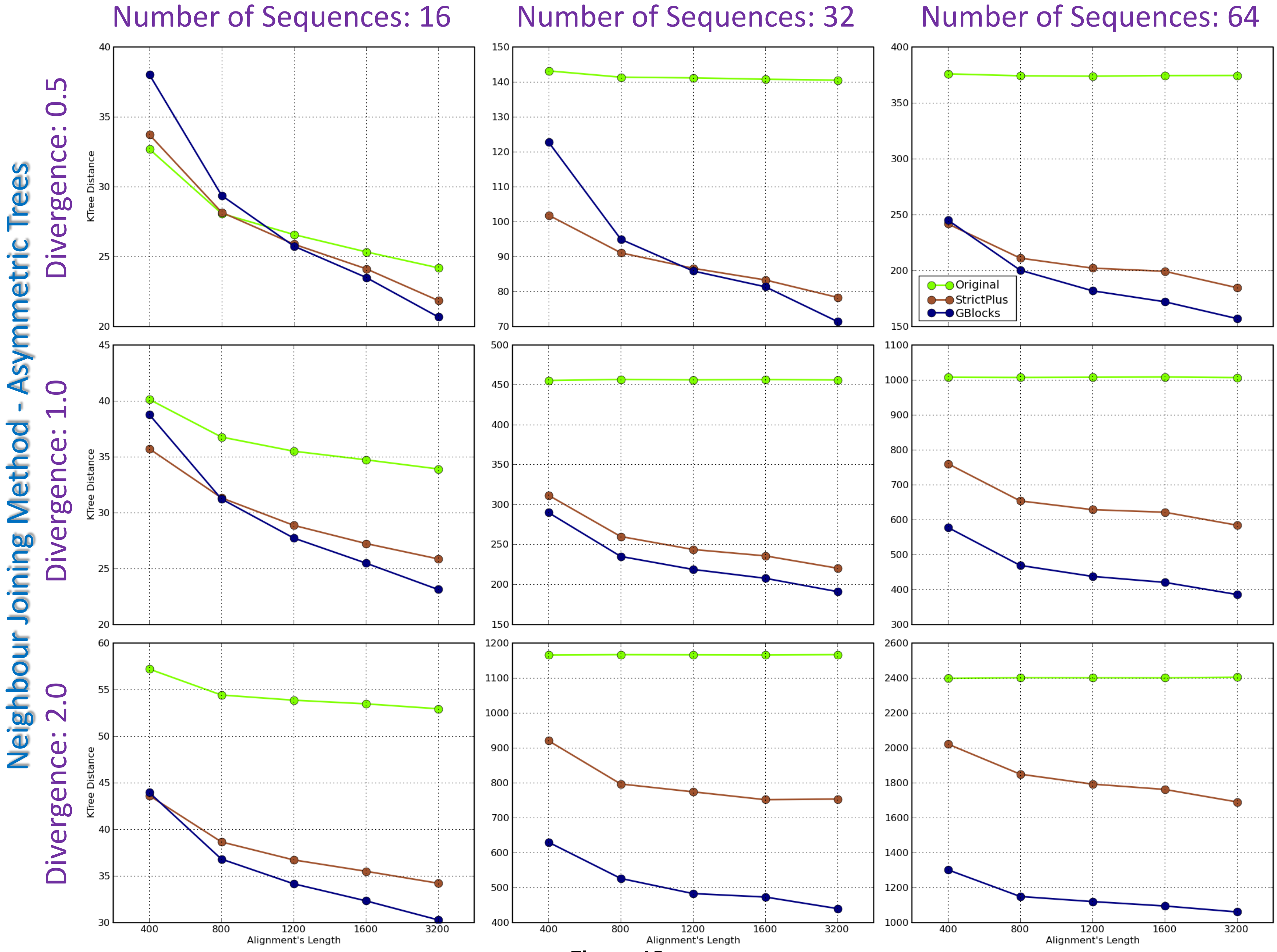

Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Maximum Likelihood Method - Symmetric Trees
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
Robinson Foulds Distance
Alignment's Length
400
800
1200
1600
3200
Original
GBlocks
Automated1

Figure 11



Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Neighbour Joining Method - Asymmetric Trees
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
KTree Distance
Alignment's Length
400
800
1200
1600
3200
Original
StrictPlus
GBlocks

Figure 12

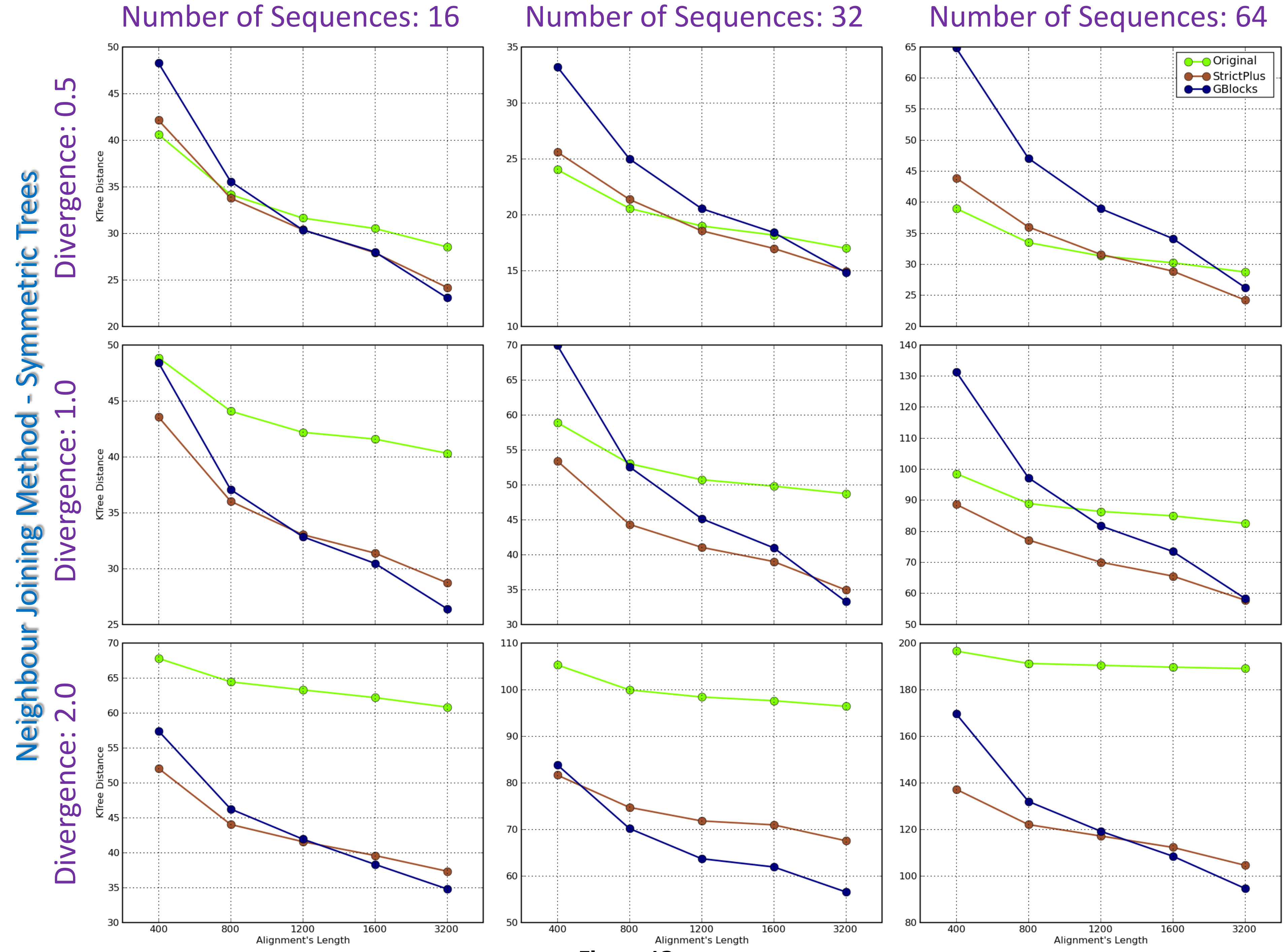
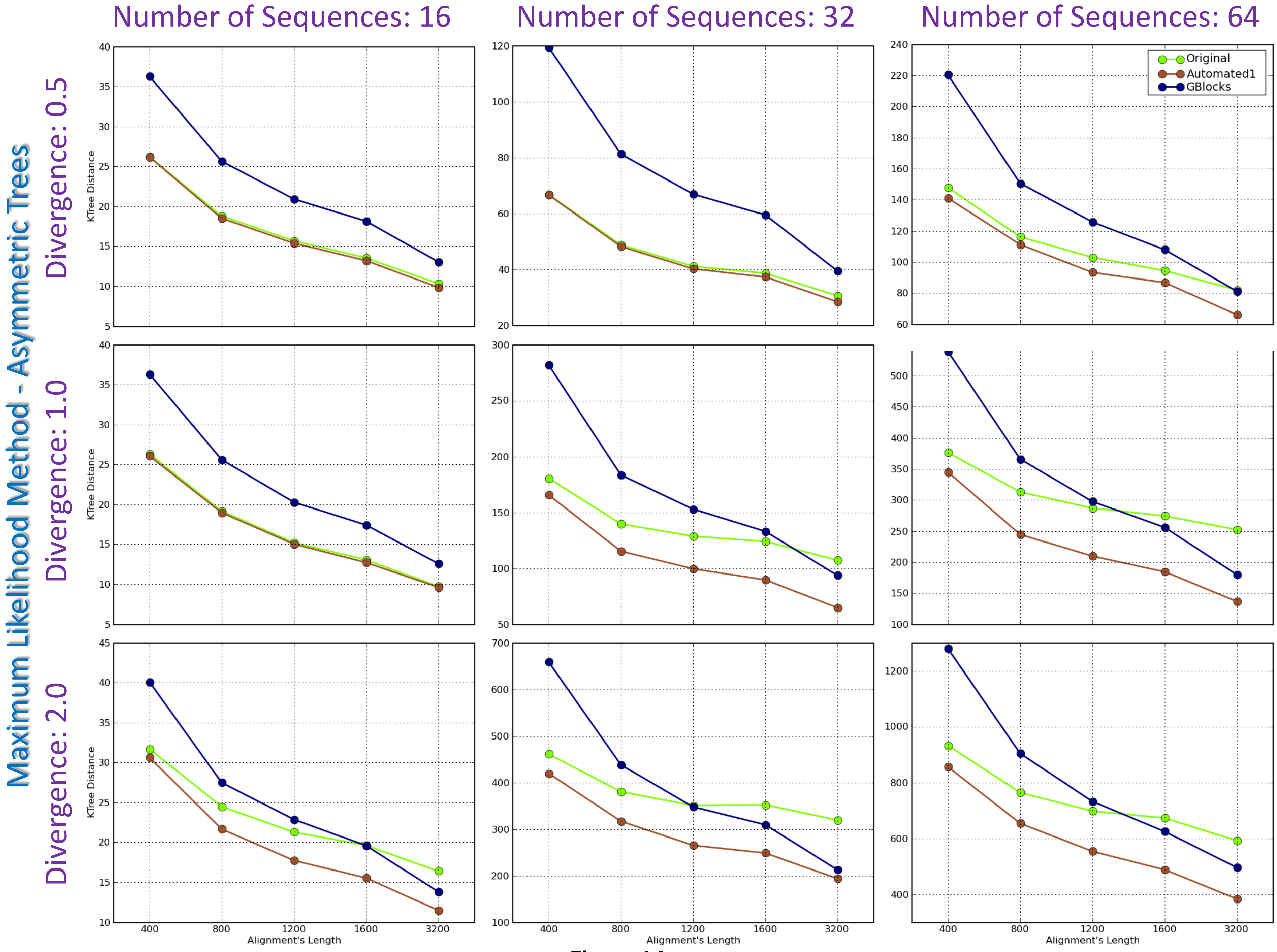

Neighbour Joining Method - Symmetric Trees
Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
KTree Distance
Alignment's Length
Original
StrictPlus
GBlocks

Figure 13



Maximum Likelihood Method - Asymmetric Trees
Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
KTree Distance
Alignment's Length
Original
Automated1
GBlocks

Figure 14

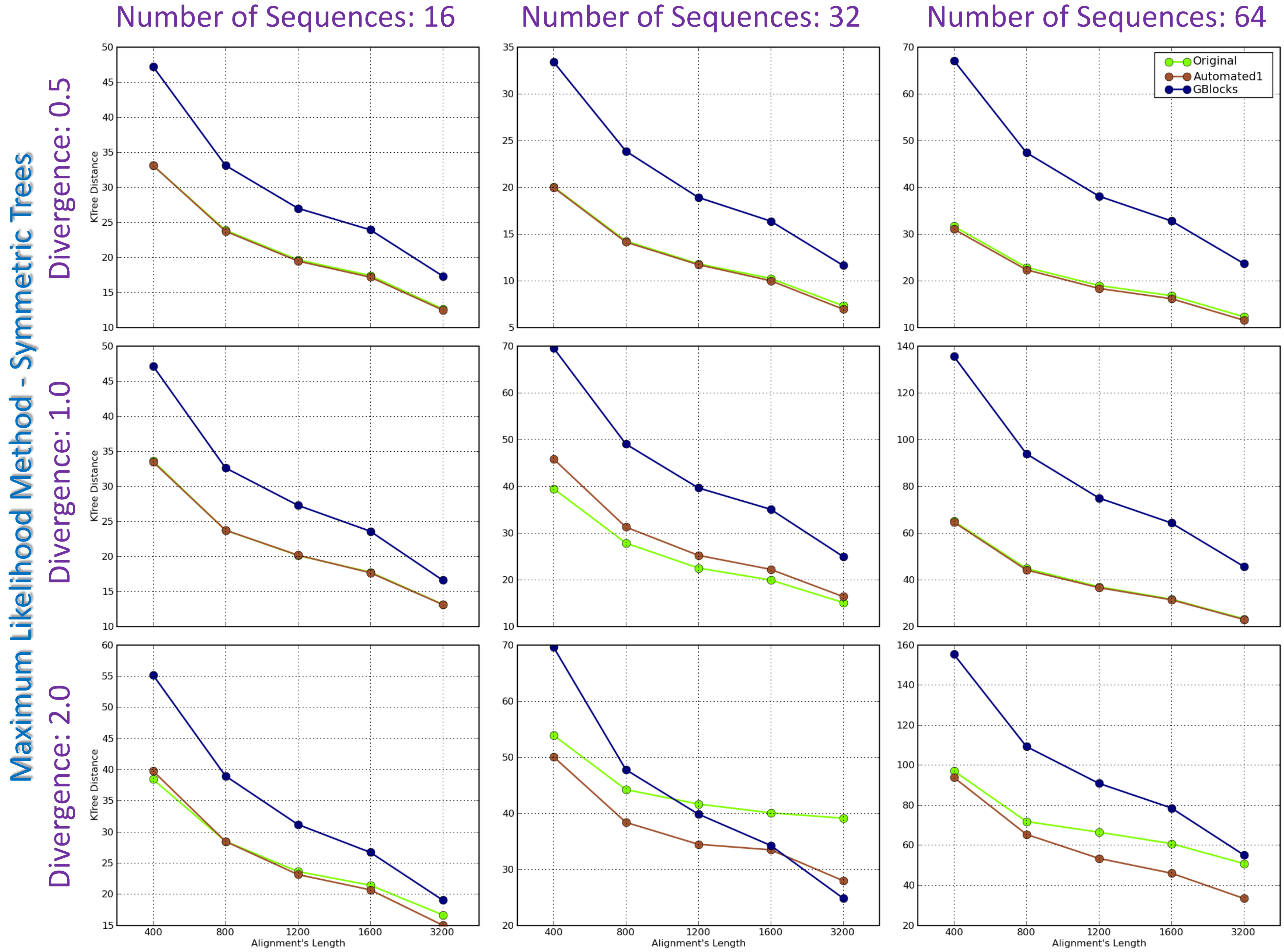

Maximum Likelihood Method - Symmetric Trees
Number of Sequences: 16
Number of Sequences: 32
Number of Sequences: 64
Divergence: 0.5
Divergence: 1.0
Divergence: 2.0
KTree Distance
Alignment's Length
400
800
1200
1600
3200
Original
Automated1
GBlocks

Figure 15