PlaceMyFossils user's guide

SCRIPT OUTLINE

PlaceMyFossils.run is a TNT script (TNT= Tree Analysis Using New Technologies, Goloboff & Morales 2023) that simplifies and extends the phylogenetic analyses using backbone trees. The script allows placing any number of species (referred to as "query species") on a backbone topology while considering one of these species as target for subsequent in-depth analyses. Phylogenetic searches can be run under three different optimality criteria: Maximum Parsimony, Implied Weighting and Maximum Likelihood. The dataset can include discrete, continuous (1-dimensional) and landmark data either alone or in combination (except for analyses under Maximum likelihood, which only allows discrete characters). The script generates a graphical representation of the results (stored as a SVG file) showing the placement of the target species on the backbone topology, coloring the branches according to the degree of optimality of the alternative positions and printing the score differences on each branch. In addition, it produces a parenthetical tree file including every optimal tree obtained. The script also allows the comparison of the results obtained under the different optimality criterion, the results are graphically displayed on the tree as a colored checkerboard on each branch with colors representing the optimality of each placement of the target species. A similar analysis can be run comparing the results under different analytical settings for a given optimality criterion (i.e. sensitivity analyses). In the case of ML analyses, the different settings imply linked-unlinked models and/or different character spaces. In the case of parsimony analysis under implied weighting (Goloboff 1993), the conditions represent different values of strength against homoplasious characters (i.e. concavity values). In addition, the script allows running user-defined sensitivity analyses by reading the different settings from a text file. These settings may specify character activity, character weights, and changing transformation costs, among others. If a sensitivity analysis is performed, the *svg* tree file generated by the script displays a colored checkerboard on each node representing the scores of placing the target species under each analytical condition. The sensitivity analysis can be combined with a Leave-One-Out Validation Approach (LOOV) in order to select the best analytical condition for subsequent placement of the query species. When character partitions are defined in the dataset file (i.e. defining character groups), it is also possible to generate an *svg* file displaying the scores of placing the target species on each branch of the tree for each character partition using a colored checkerboard similar to that used in the sensitivity approach. The script also allows evaluating the error in placing each species included in the backbone topology with the idea of approximating the error associated with the placement obtained for the query species. The error is displayed for each subtree, and can be calculated either for the complete dataset or for each character partition. LOOV and error analyses can be run considering the complete character sampling or restricted to the characters scored for the target species. Finally, the script allows generating a chart in *svg* format showing the scores for each character (or character partitions) as different colors when the target species is placed on each possible branch of the tree. A text file including character and character group scores for each possible position of the target species are also generated. *PlaceMyFossils* is mainly aimed at dealing with a single reference topology. However, it also implements some basic functions to deal with multiple reference trees. In addition, the script includes a function to allow the target species to modify the reference tree. Before using the script, it is strongly advisable to read the paper describing it (Catalano et al. 2024).

The script can be run in the GUI (Graphic Unit Interface) versions of TNT for Windows, Linux and Mac operating systems. TNT can be freely downloaded from https://www.lillo.org.ar/phylogeny/tnt. The custom version of *PlaceMyFossils* can handle datasets with up to 1000 taxa and 1000 characters. Please contact the author for a version that can handle larger datasets.

Important: please check that you have the latest TNT version installed in your computer.

The script can be downloaded from:
https://www.lillo.org.ar/phylogeny/tnt/scripts/PlaceMyFossils_script.zip

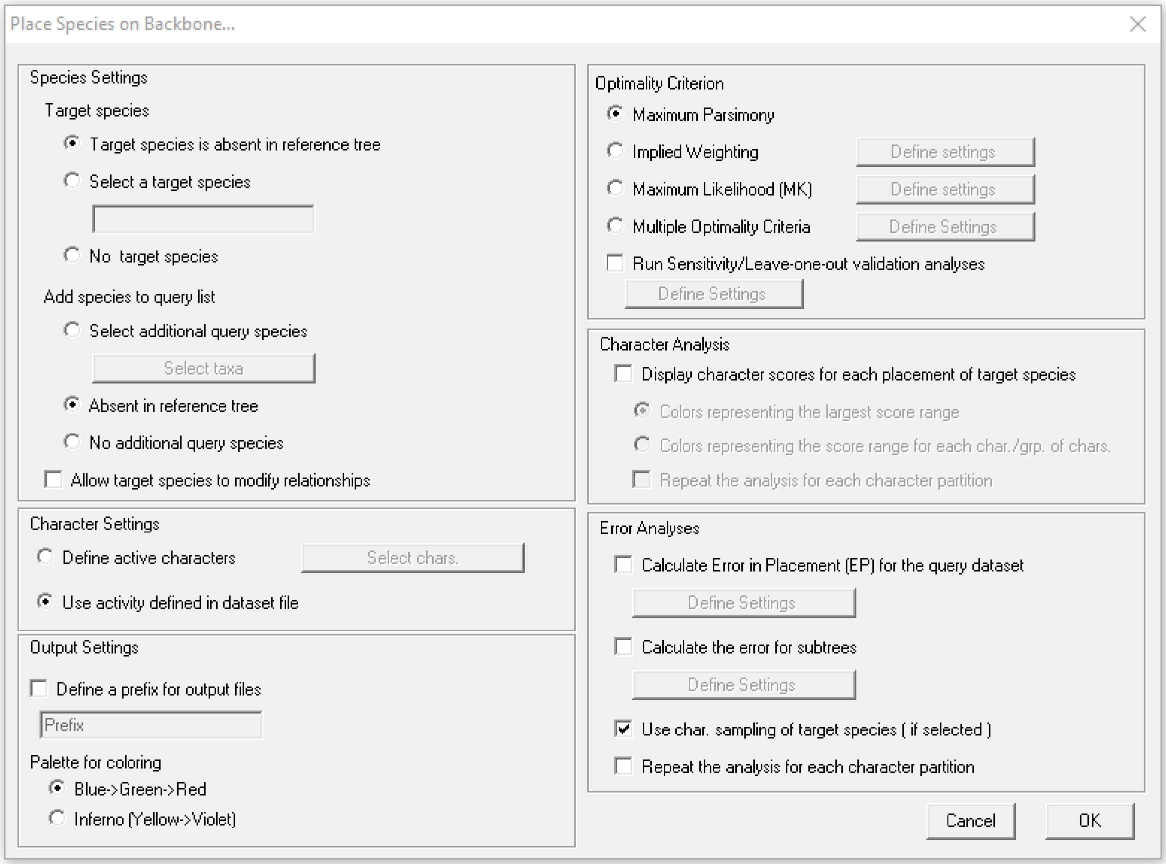
The correct citation for the script is:
Catalano SA, Escapa I, Pugh KD, Goloboff PA, Hammond AS, S Almécija (2024) *PlaceMyFossils*: An integrative approach to analyze and visualize the phylogenetic placement of fossils using backbone trees. Systematic Biology.

In addition, the citation for TNT is:
Goloboff, PA, & Morales, M.E. (2023). TNT version 1.6, with a graphical interface for MacOS and Linux, including new routines in parallel. Cladistics, 39(2), 144-153.

Please contact Santiago Catalano (sacatalano@gmail.com) for bug reports or assistance.

INTERACTING WITH THE SCRIPT

The interaction with the script is by means of dialogs. Once the script is read in TNT (**File/Open Input file**) a window pops-out to select the dataset file (see "**Input Files**" section. After selecting the dataset file, a second window pops-out to select the reference/backbone tree (see "**Input File**" Section). If both files are correctly read, a dialog is displayed to choose the conditions and analyses to be performed (Fig. 1). The dialog displayed depends on the number of reference trees included in the tree file, showing a dialog with limited number of functions for the case of multiple trees.


Place Species on Backbone...

Species Settings

Target species

[✓ Target species is absent in reference tree]

Select a target species

No target species

Add species to query list

Select additional query species

Select taxa

[✓ Absent in reference tree]

No additional query species

Allow target species to modify relationships

Character Settings

Define active characters — Select chars.

[✓ Use activity defined in dataset file]

Output Settings

Define a prefix for output files

Prefix

Palette for coloring

[✓ Blue->Green->Red]

Inferno (Yellow->Violet)

Optimality Criterion

[✓ Maximum Parsimony]

Implied Weighting — Define settings

Maximum Likelihood (MK) — Define settings

Multiple Optimality Criteria — Define Settings

Run Sensitivity/Leave-one-out validation analyses

Define Settings

Character Analysis

Display character scores for each placement of target species

[✓ Colors representing the largest score range]

Colors representing the score range for each char./grp. of chars.

Repeat the analysis for each character partition

Error Analyses

Calculate Error in Placement (EP) for the query dataset

Define Settings

Calculate the error for subtrees

Define Settings

[✓ Use char. sampling of target species (if selected)]

Repeat the analysis for each character partition

Cancel — OK

Figure 1. Main dialog of *PlaceMyFossils* script. This dialog is displayed when a single reference tree is included in the tree file. A dialog with reduced options is displayed when multiple trees are included in the input tree file (see Fig. 4).

Important: Neither the path nor the file names can include blank spaces.

QUICK ANALYSIS

The simplest analysis consists of placing a single query species (target species) on a reference tree in order to determine the optimal placement and the sub-optimality of alternative placements. This analysis is run in five simple steps:

1-. Read the script in TNT. Go to **File/OpenInputFile**, choose “All files” as the extension and click on *PlaceMyFossils.run*

2-. Choose the dataset file (must have “tnt” extension)

3-. Choose the tree file, including one single reference tree (must have “tre” extension)

4--. If the target species is not included in the reference tree skip this step. If the target species is included in the reference tree click on “Select a target species”, and write its full name in the box below.

5.- Press the button “OK” at the bottom-right corner of the dialog.

This quick analysis will place the query species in its optimal placement(s) under the Maximum Parsimony criterion, producing two different outputs. 1- An *svg* file (Query_placement.svg) showing the optimal position and the scores for alternative suboptimal placements. 2- A tree file in parenthetical notation where the target species is placed in the optimal placement.

Important: All the species in the data matrix should be active. If the user wants to exclude one or more species from the analyses, these should be excluded from the data matrix beforehand.

Important: Every species in the reference tree should be in the dataset. On the contrary, TNT will return an error message.

Important: The dataset file and the tree file should include the string *proc/*; at the end of the file.

DIALOG DESCRIPTION: RUNNING A SINGLE REFERENCE TREE

“Species settings” box

By default, if a single taxon is absent from the reference tree but present in the dataset, it will be considered as the target species. The target species is the species that will be the reference for the analyses. If there is more than one species absent from the reference tree, hence it is required to choose one of them as target species. If the reference tree is complete, you can choose the target species by clicking “***Select a target species***” and writing its full name in the box below. By clicking on “***No target species***”, the script will only place the query species on the tree, with no further analysis. In addition to the target, additional query species can be included in the analysis. These can be represented by the species absent in the reference tree or can be selected by clicking on “***Select additional query species***”, then press “***Select taxa***” button, and then choose the species from the list displayed.

In the “Species Settings” box it is also possible to choose the option “***Allow target species to modify relationships”***. This option is aimed at considering the cases where the inclusion of the target species may modify the placement of the rest of the species. In this analysis the reference tree should have been obtained including the target species in an unconstrained analysis beforehand. As in traditional backbone-based analysis, this function calculates the score of placing the target species on each node of the reference tree. However, every time this species is placed on a branch to calculate its score, the rest of the species (the “backbone species”) are allowed to change its placement so that the score is minimized. The species are not allowed to change to any placement: the composition of the splits defined by the branch where the target species is temporarily placed is to calculate the score is conserved. The approach visits each node of the reference tree (with the target species being pruned). On each node the target species is grafted, the script defines two constrains, one including all the species that descend from the node, and a second including the same species plus the target species. Hence the constrains are defined so the relative placement of all the species at both ends of the branch where the target species is grafted, remains the same. Afterwards, a search is run under the constrains defined. The score obtained in that search represents the score of the placement of the target species on that branch. This is repeated node by node to obtain the score of placing the target species at each node of the tree. This approach is probably the most useful for datasets where the reference topology is obtained from morphological data, and the user has the suspicion that the inclusion of the fragmentary query species may alter the relationships of the rest of the species. If scores calculated along the tree are coincident when running the analysis with the backbone fixed (i.e. not allowing the query species to modify relationships), the user can be confident in running the rest of the analyses (e.g. error analysis, character analysis) considering the reference topology as fixed.

"Output settings" box

In this box it is possible to define a prefix, so the name of each output file generated will start with the prefix defined. In addition, it is possible to change the palette for displaying the values on the tree.

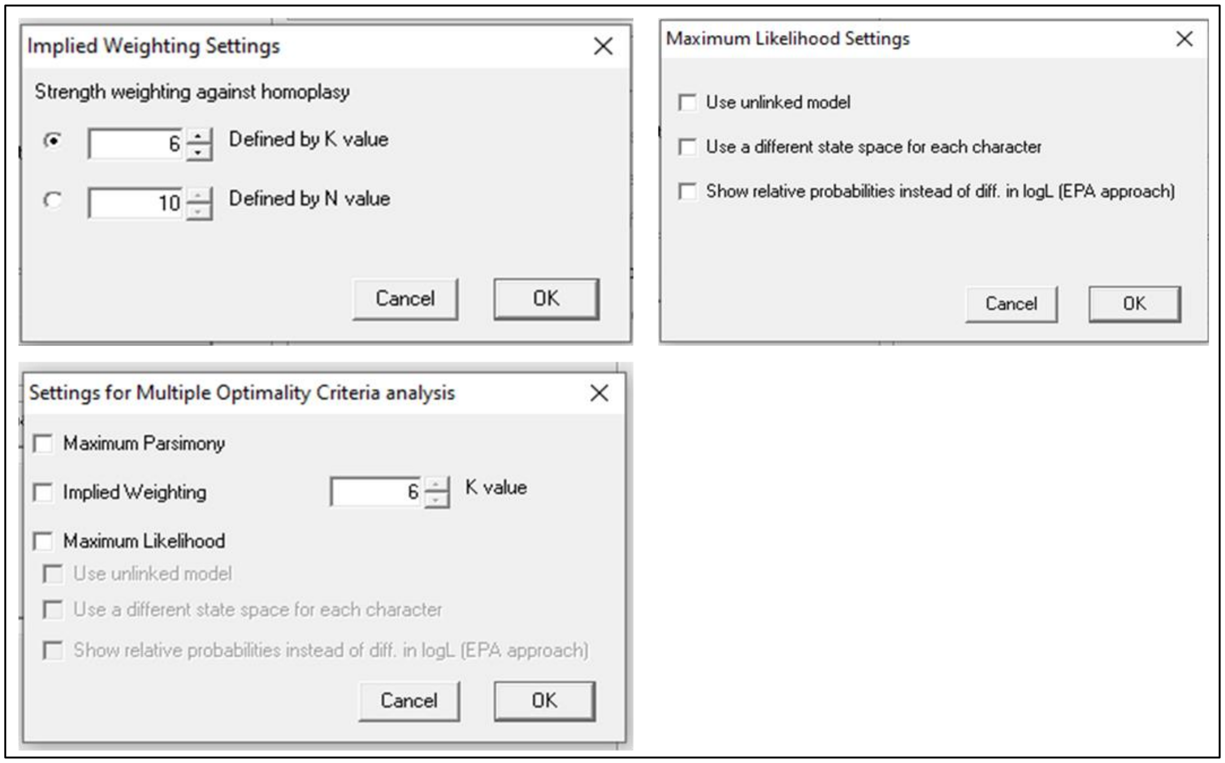
Important: Do not leave a blank space after the prefix.

"Optimality criteria" box

Species included in the query list can be placed on the backbone topology using different optimality criteria: Maximum Parsimony (MP), Implied Weighting (IW) and Maximum Likelihood (ML). While MP and IW can be chosen to analyze discrete, 1-d continuous and landmark characters, ML is only currently available for discrete characters. Landmark analysis is run considering the approach proposed by Catalano et al. (2010) and the algorithms described in Goloboff & Catalano (2011,2016). Landmark analyses are run with the default TNT settings. Alternatively, the user can include different settings in the dataset file. 1D continuous characters are analyzed using the algorithms proposed by Goloboff et al. (2006). Continuous characters are not standardized by the script, but the commands to do that can be included in the datafile following the datamatrix. A detailed guide (Catalano & Goloboff 2019) for the use of continuous 1-D and landmark characters in TNT is available here. The extended implied weighting (Goloboff 2014), can also be run by including the conditions in the settings file and running it as a sensitivity analysis (see below).

Maximum likelihood analysis considers a Markov model as implemented in the function *mklik* of TNT. Branch lengths for the ML score calculation are determined from the query dataset, not from the branch length of the reference tree. The score in ML analyses is by default expressed as -*log*L values. Alternatively, it is possible to express the values as likelihood weight ratios (LWR) as implemented in the Evolutionary Placement Algorithm (EPA; Berger et al. 2011). LWR for a given branch is calculated as the likelihood of placing the target species in that branch, divided by the sum of the likelihood for all possible placements. While the ranking of branches according to the sub-optimality does not vary whether the -*log*L or the LWR are considered (except that the best scores are now the highest instead of the lowest values), the distribution of values is generally very different with one or a few branches with the best scores—high probability values—and many branches with very low values. The score under Markov (MK) model for each character/ partition will depend on the settings defined in the ML analysis. If the analysis is run using the linked model, the score for each character/partition will be calculated considering the branch length for the complete dataset. For the unlinked model the score for each character partition will be calculated considering its own branch length. The scores calculated under MK are not modified by character weights. ML criterion is not available for the following analyses: multiple query species search, error in placement and leave-one-out validation (Table 2).

Different settings can be selected for each optimality criterion, by clicking on the “Define Settings" buttons. For IW, the weighting strength against homoplasious characters can be defined by choosing a particular concavity value (k). Alternatively, it is possible to choose a N value that represents the maximum range of implied weights. This value is internally translated to a K value (see Goloboff et al., 2008a, Torres et al. 2022). For ML analysis it is possible to choose the type of state space considered to calculate the scores. A global morphospace implies using the largest number of states observed in a character as the space size for every character in the datamatrix. Alternatively, no-global morphospace means that the number of possible states for each character is the observed one. If characters are assigned to different groups in the dataset file (using the command *xgroup*), it also is possible to calculate ML scores using an unlinked model.


Implied Weighting Settings

Strength weighting against homoplasy

[✓ 6 Defined by K value]

10 Defined by N value

Maximum Likelihood Settings

☐ Use unlinked model

☐ Use a different state space for each character

☐ Show relative probabilities instead of diff. in logL (EPA approach)

Settings for Multiple Optimality Criteria analysis

☐ Maximum Parsimony

☐ Implied Weighting 6 K value

☐ Maximum Likelihood

☐ Use unlinked model

☐ Use a different state space for each character

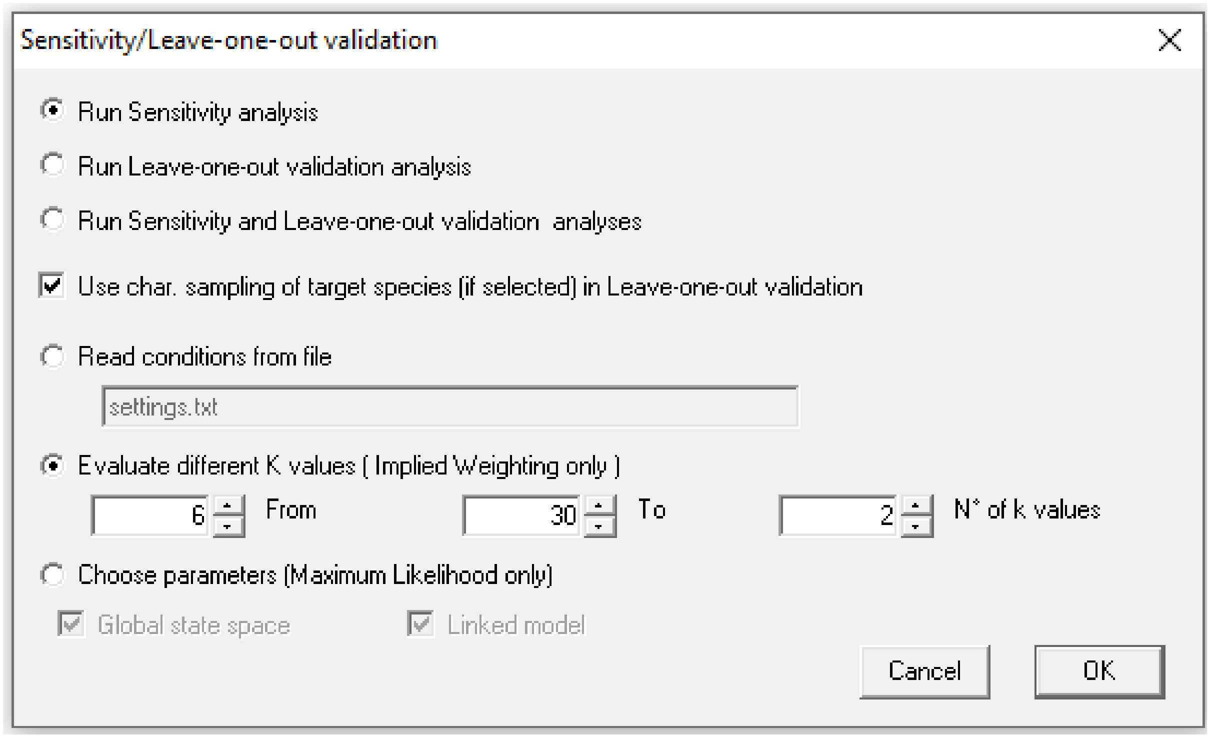
☐ Show relative probabilities instead of diff. in logL (EPA approach)

Figure 2. Sub-dialogs to define the settings for each optimality criterion when run individually (upper row), or when running multiple optimality criteria analysis (lower row).

When the query list includes a single species, the optimal placement is defined by checking the score of each possible placement, hence implying an exact constrained search. When the query list includes more than one species, the species are placed by means of a branch-swapping procedure using the backbone topology as a tree constraint.

Another option included in the optimality criterion box is to run an analysis that compares the placement of the target species under the different optimality criterion ("***Multiple Optimality Criteria***" option). If a target species is selected, the script generates an *svg* tree file that includes a colored checkerboard on each node showing the scores (as different colors) of placing the target species under each optimality criterion (see Output section) and the optimal placements. The actual scores are not shown in this analysis. These can be obtained by running individual analyses for each optimality criterion. In addition, a parenthetical tree file including the placement of the query species is generated for each optimality criterion chosen.

The script implements sensitivity and Leave-One-Out Validation (LOOV) analyses. The sensitivity analysis allows the user to assess how different analytical conditions affect the position of the query species and the position and optimality scores of the target species. In the leave-one-out validation approach the script selects, among the different analytical conditions, the one that minimizes the error in placement of the backbone species and uses those settings to place the query species on the tree. Once selected by clicking on "***Run Sensitivity and leave-one-out validation approaches***", the particular analysis to be conducted and the options are set on a sub-dialogs accessed by clicking on the *"**Define Settings***" button (Fig. 3) The first option of this sub-dialogs allows the user to choose between running a sensitivity analysis, a leave-one-out validation analysis or both analyses simultaneously. The analytical settings considered depend on the particular optimality criterion chosen. For IW it is possible to choose different concavity values. The user can define a minimum and maximum value for the concavity constant and the number of intermediate values to run the analysis. For ML the different conditions that can be evaluated are linked/unlinked models and (no)global state spaces. In addition, it is also possible to run the sensitivity/leave-one-out validation analyses by reading the analytical conditions from a text file (see Input Files section).


Sensitivity/Leave-one-out validation

● Run Sensitivity analysis

○ Run Leave-one-out validation analysis

○ Run Sensitivity and Leave-one-out validation analyses

✓ Use char. sampling of target species (if selected) in Leave-one-out validation

○ Read conditions from file

settings.txt

● Evaluate different K values (Implied Weighting only)

6 From 30 To 2 N° of k values

○ Choose parameters (Maximum Likelihood only)

✓ Global state space ✓ Linked model

Cancel OK

Figure 3. Dialog to define the options for the Sensitivity and leave-one-out validation approach.

The script also allows running a leave-one out validation analysis to select the best the analytical conditions to place the target species. The leave-one-out validation approach selects the analytical conditions that minimizes the error in placement for the backbone species (See Catalano et al. 2024 for further detail). As in the sensitivity analysis, the conditions can either be selected from the dialog (for IW and ML), or be read from a text file, including the different settings for any of the three optimality criteria (Fig. 4). If the leave-one-out validation analysis selects more than one condition as optimal, one of them (the first) is chosen and the subsequent analyses are run considering that condition. The user may decide to run additional analyses considering the rest of the optimal conditions one by one afterward. The script also has the option of running both sensitivity and leave-one-out validation analyses combined. In this case the scores for the target species are shown on the tree for every condition using the colored checkerboard, while highlighting the condition selected by the leave-one-out validation approach. It is possible to run the leave-one-out validation analysis considering the character sampling of the target species. This is a very useful option because the best condition selected when taking into account the complete character sampling need not to be the same as the condition that best performs for a possibly limited character sampling present in the target species. This option is selected by checking the box ***"Use char. sampling of target species (if selected), in leave-one-out validation approach***".

The sensitivity analysis produces two types of output files. First, it produces one parenthetical tree file per analytical condition, where the query species is included in the optimal place. This is the only output produced when no target species is selected. If a target species is selected, the script generates an SVG tree file that includes a colored checkerboard on each node showing the scores (as different colors) of placing the target species under each analytical condition (see Output section). The optimal placement for each condition is indicated with an asterisk.

"Character settings" box

In addition to the settings defined in the dataset file, it is possible to select from the dialog which characters will be considered for the analysis (i.e. character activity). If the datafile includes commands for character activity, these are maintained along analyses. However, if characters are selected from the dialog by selecting the option "***Define character activity***" and clicking on *"**Select characters**"*), the activity definitions included in the file are overridden and the analysis will only consider the character activity defined by the user from the dialog. In sensitivity/leave-one-out validation approaches, the activity will be reset to the original values after each iteration.

When the leave-one-out validation analysis is run considering the character sampling of the target species (by checking the box "***Use char. sampling of target species (if selected), in leave-one-out validation***"), character selection defined in the dataset file or from the dialog is respected. This means that active characters will be those that are present in the target species and that were part of the list of active characters.

"Character analyses" box

Character analysis is only available when a single optimality criterion is chosen. It is not available for "***Multiple optimality criterion***" or "***Sensitivity/leave-one-out validation***" options. The character analysis can be selected by checking the option "***Generate char w/ char. scores for each placement of target species***". This analysis allows to visualize the score of the target species for each character and (if defined) each character partition, on each branch of the tree. The character partitions are defined in the dataset file using the command *xgroup (see "Input Files"* section*)*. For each partition, the score is determined as the sum of the scores for its characters. The analysis for character partitions can be selected by checking the option "***Repeat the analysis for each character partition***". The script generates a chart in an *svg* format (*Chars_score_table.svg* and *Partitions_score_table.svg*) showing as different colors the increase/decrease of scores for each character or character partition when the target species is placed on each possible branch of the tree. Text files (*Chars_score_table.out* and *Partitions_score_table.out)* including the numerical values used in the colored table are also generated. If the option "***Repeat the analysis for each character partition***" is checked, the script also generates an *svg* file (*Query_placement_partitions.svg*) with a graphical representation of the optimality of the placement of the target species on each node for each partition, using a colored checkerboard (Fig. 3d).

When generating the chart and tree showing the scores for each placement of the target species for each character and /or group of characters, it is possible to choose two different color representations. One possibility is to use the same color gradient for all the characters/groups of characters (option "***Colors representing the largest score range***".) This means that the span of the color gradient will be defined by the character/ partition that presents the largest span of scores. This option is very useful to quickly determine which characters/character partitions are more decisive in placing the target species on the tree. When this option is selected, the bar showing the color gradient indicates the difference in scores. Alternatively, (Option "***Colors representing the score range for each char/grp. of chars.***"), it is also possible to use a different color gradient for each character/group of characters so each of them will include the two extreme colors of the palette. Notice that in this case the difference in color may represent different ranges of scores (as a consequence the bar does not show a scale of scores over it). This option is useful when the focus is on analyzing each character partition individually with no direct interest in comparing the scores of each group of characters.

When analyzing multiple query species, the score is calculated by leaving the target species in a particular position while the rest of the species are placed so that the score is minimum by means of a branch swapping procedure. This is repeated at each branch of the tree. This implies that the rest of the query species can be placed in more than one optimal placement. For the complete dataset this does not matter. The score calculated for the target species is the same whether the rest of the species present one or multiple optimal placements. However, the presence of alternative optimal placements for the rest of the query species may produce alternative scores for each character/partition, and hence it cannot be displayed with a simple color code. As a consequence, character/partition analysis can only be conducted when a single query species (the target species) is analyzed.

"Error analyses" box

The script implements a novel approach to determine the error in placing the query species on the backbone topology. The approach is described in detail in Catalano et al. (2024) and the corresponding supplementary material. The analysis is conducted by pruning each of the backbone species and recording the optimal placement of each of the species considering the query dataset. From that information two different sorts of errors are calculated. One of them (Error in placement; *EP*) calculates the error associated with the datasets (complete dataset or each character partition). The other error (In/Out subtree error) is calculated for each subtree of the reference tree. *EP* values are calculated as the sum of the nodal distance between the correct placement of the backbone species (i.e. original placement in the reference topology) and the optimal position retrieved in each case. This analysis is repeated by considering the different character partitions. The information generated in this analysis is included in three different files. The file "*Errors_by_partition.out*" includes the error in placement for the complete dataset and each character partition. The raw data considered to calculate this error (the nodal distance for each species) on each partition is also printed. In the presence of multiple optimal placement, the nodal distance calculated for each backbone species is not unique. Hence, user can choose among calculating the EP from the minimum, the maximum and the mean nodal distance for each species. In addition, the file includes information about the character and taxon sampling considered to calculate the error in each partition. A second file named "*Errors_by_partition.svg*" shows as a barplot the EP value by character partition (if defined) and for the complete dataset. It is possible also to calculate a probability associated to the error in placement.

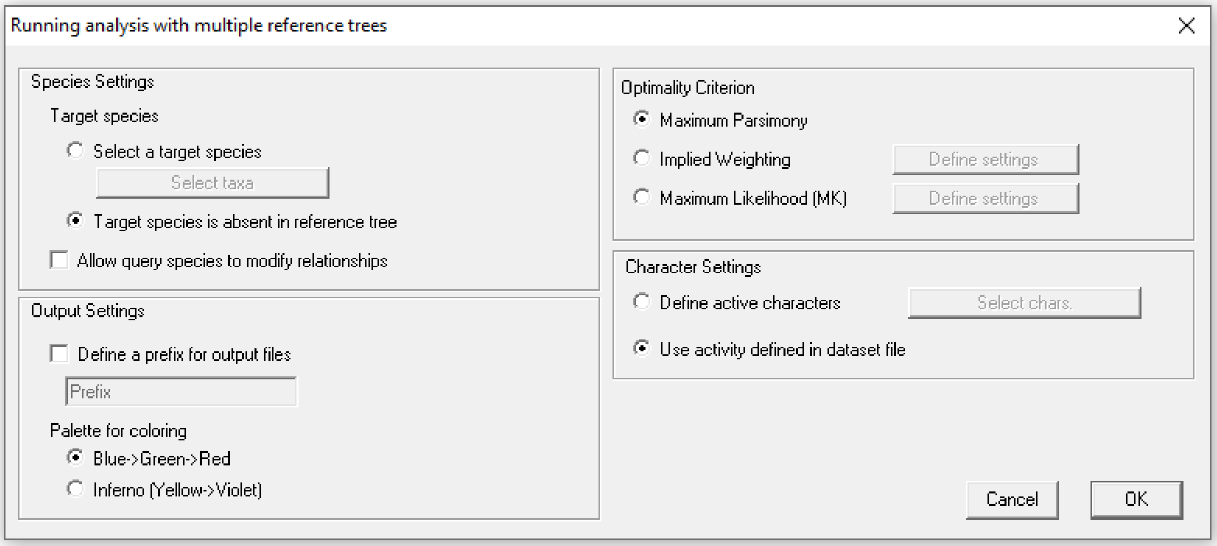
The second type of error is calculated for each subtree. The basic information to calculate this error is the same (the optimal placement of each backbone species). However, in this case what is calculated is how often species that are not part of the subtree are placed in it (in-subtree error - ISE) and how common species that are part of a subtree are placed in a different clade (out-subtree error -OSE). The error values displayed represent the percentage of the maximum possible error that can be calculated for a given subtree. As in the case of *EP* calculation, *ISE*, and *OSE* are affected by how ambiguity in placement (i.e., multiple optimal placements for each backbone species) is considered. This means that, in some cases, one backbone species can have an optimal placement within the subtree, and an alternative placement outside the subtree. *PlaceMyFossils* has the option of excluding from the error calculation those cases where the placement of a backbone species includes its correct (original) placement as one of the optimal placements. In that case, the error does not include the ambiguity.

When the error analysis is selected, the *ISE* value is displayed as a horizontal bar in the subtending branch of each subtree in the *Query_placement.svg* file (See Output section). When the error is calculated for different character partitions (by checking the option "***Repeat the analysis for each character partition"***), a new *svg* file is generated ("*Errors_by subtree.svg*") showing the *ISE* for each partition using a colored checkerboard on each node. In addition to the graphical display of the errors, the script also generates a text file including the ISE/*OSE* values for each dataset (complete dataset and character partitions; "*Errors_by_subtree.out*"), for each subtree of the backbone tree. As in the case of the leave-one-out validation approach, the error analysis can also be run considering the character sampling of the target species. This is chosen by checking "***Use char. sampling of target species (if selected)***" in the "*Error analysis*" box.

DIALOG DESCRIPTION- RUNNING WITH MULTIPLE REFERENCE TREE

When multiple reference trees are included in the tree file, a reduced dialog is displayed. This is because only basic functions are included when dealing with multiple trees, and only a single query species can be placed in the backbone topology. To deal with multiple trees, we have included a function similar to that proposed by Klopfstein & Spasojevic (2019) displaying and summarizing the scores of alternative reference trees on the branches of a summary tree. In our approach the script calculates the score of placing the target species on each branch of each backbone tree and displays these values on the strict consensus of all backbone trees having previously pruned the target species. The scores (displayed as numbers and colors) represent the minimum score for that branch in all the reference trees. In addition, for those branches that are not presented in the strict consensus, the minimum score is displayed on the most recent common ancestor that is present in the strict consensus, by coloring the vertical line as in

Klopfstein & Spasojevic (2019) approach (see section **Output files**). As in the case of working with a single reference tree, it is possible to let the target species modify the reference tree. When working with multiple reference trees, the trees are re-rooted on the outgroup species. It is not possible to be rooted on an internal node because these differ among trees. To root the tree in a different terminal node, the user need to change the outgroup species either by placing it as the first taxon in the datamatrix, or by including the command *outgroup* in the data file, after the dataset.



Running analysis with multiple reference trees

Species Settings

Target species

Select a target species

[✓ Target species is absent in reference tree]

Allow query species to modify relationships

Output Settings

Define a prefix for output files

Palette for coloring

[✓ Blue->Green->Red]

Inferno (Yellow->Violet)

Optimality Criterion

[✓ Maximum Parsimony]

Implied Weighting

Maximum Likelihood (MK)

Character Settings

Define active characters

[✓ Use activity defined in dataset file]

Figure 4. Main dialog for the case of having multiple reference trees.

INPUT FILES

The script requires two files: (i) a file including the dataset and (ii) a tree file in parenthetical notation.

Dataset file

In addition to the character matrix in TNT format, the dataset file may include different settings such as character weights, additivity, activity, etc. It is in this file that groups of characters (partitions) can be defined. These character partitions should be defined in the dataset file using the TNT command *xgroup*. The following example shows how to define the groups of characters.

```
xgroup = 0 (leaf)     0.23  ;
xgroup = 1 (root)     24.47 ;
xgroup = 2 (seed)     50.61 ;
xgroup = 3 (flower) 62.73 ;
```

Note: The dataset file should always end with a line containing the string **proc/;** (the TNT command that indicates the end of the datafile). Do not include the backbone topology in the dataset file. It should be included in a different file. Do not include blank spaces in the dataset file name.

Tree file

The script reads the backbone topology from a tree file in TNT parenthetical notation. Important: All the analyses run in the script require having dichotomous trees. If the input tree has polytomies these will be randomly resolved, and a warning messaged will be displayed.

Settings file (Optional)

As previously indicated it is possible to read the settings for the sensitivity and leave-one-out validation analyses from a text file (Fig. 5). Each line in the settings file represents a condition to be run. The settings are defined using TNT commands. The file can contain up to 15 different conditions. Each condition should be included as a different line in the file (i.e. conditions are separated by a carriage return character). The conditions, in general, will imply character settings defined by the command *ccode*, but other settings can also be included. For instance, landmark settings and, continuous character standardization, etc. There can be any number of settings defined for each analytical condition as long as these are in the same line. The character settings defined in the dataset using *ccode* will be restored to the settings defined in the dataset file after each iteration. Other conditions, such as landmark settings or changes in continuous character scaling, will require to be restored including the corresponding settings in the line. An empty line should be included after the last analytical setting. The definition of character partitions cannot be changed from this file.

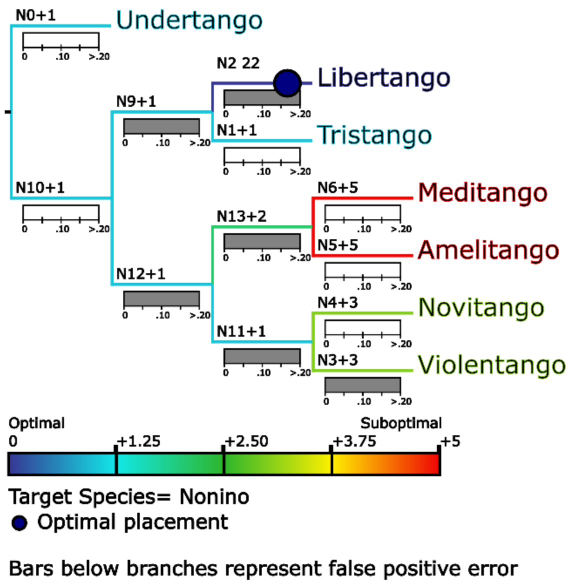
```
ccode - . ; cc + 0 1 5 ;
ccode ] . ; cc [ 0.56 ;
ccode ] . ; cc [ 0.5 ;
ccode ] . ; cc /3  6.56 ;
nstates stand 1 ;
xpiwe=; xpiwe  ]  ;
```

Figure 5. An example of a setting file that includes the alternative conditions to run a user defined sensitivity or leave-one-out validation analyses. Each line represents a different analytical condition, hence the analysis will be repeated 6 times each with a different analytical condition. An empty line should be included after the last analytical setting. Notice that in this latter case before opening the script it is necessary to turn implied weighting on (with the command piwe=;. This is defined in the TNT menu at Settings/ImpliedWeighting/BasicSettings).

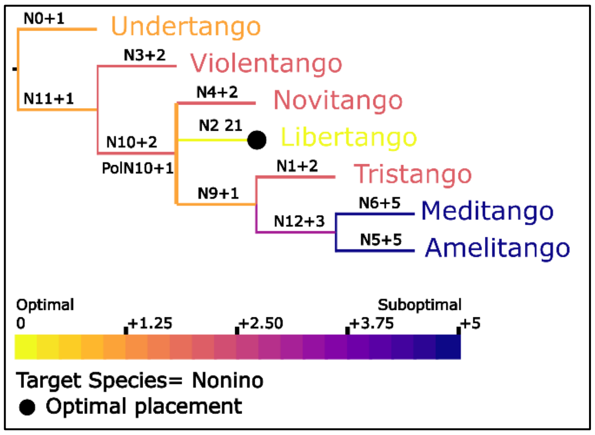
OUTPUT FILES

Query_placement.svg

The main output generated by the script is a graphical representation of the backbone tree showing the optimal position(s) of the target species, representing the scores with a color code, and also displaying the difference in score between the optimal position and each alternative suboptimal position. If the subtree error analysis is selected, this file also displays *ISE* value on each node as a horizontal bar.

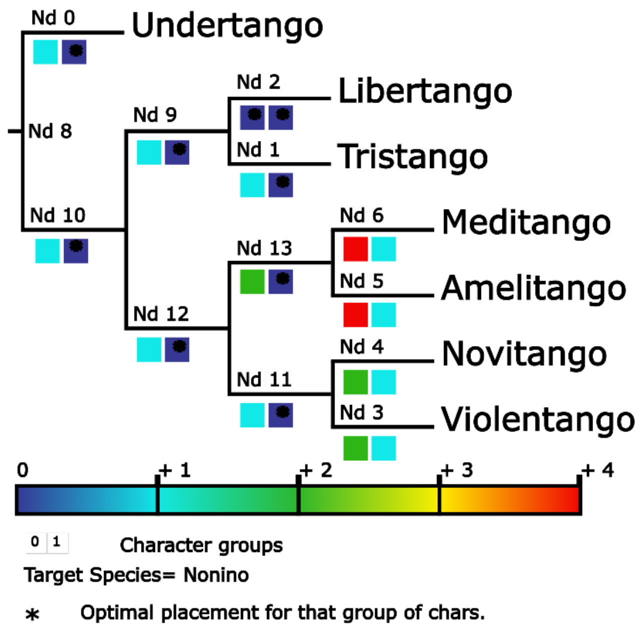


In the case of having multiple optimal trees, the scores are displayed on a strict consensus tree**.** Notice that the polytomic node has two tags. The tag above the branch represents the score obtained by placing the target species on that branch, while the tag below the branch represent the minimum score obtained for all the nodes that are collapsed into that polytomy.



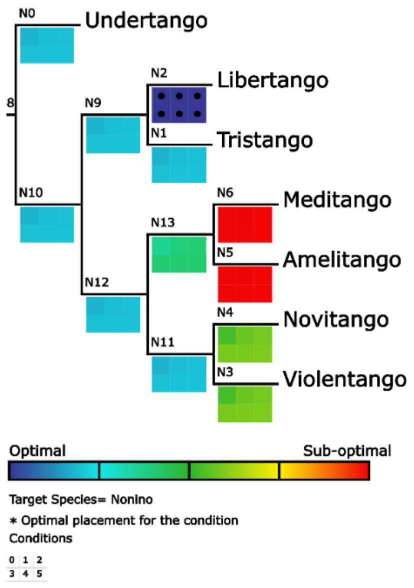
Query_placement_partitions.svg

An *svg* file with a tree showing the scores for each partition as a colored checkerboard. The best placement(s) for each partition is indicated with an asterisk.



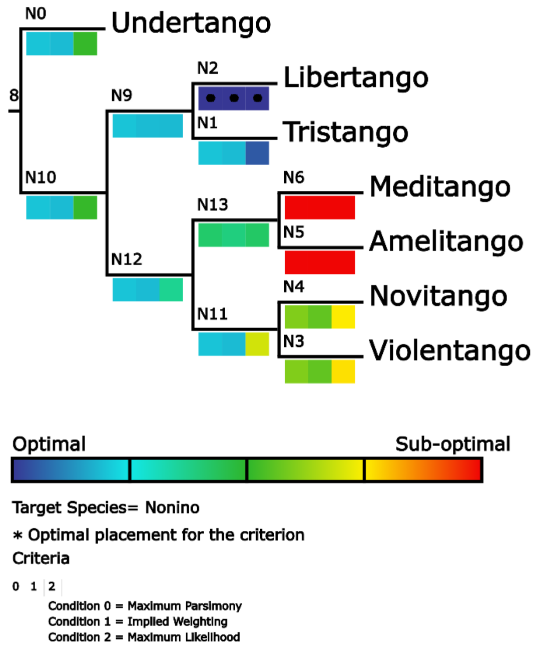
Query_sensit.svg

Displays the results of the sensitivity and leave-one-out validation analysis. It uses a colored checkerboard showing the score for each condition, and an asterisk to indicate the best placement(s) under each condition. A bar showing the color scale is included in the *svg* file. Color code in this case only represent the optimality in a relative way, because the difference in score in absolute values depend on the condition used to run each iteration



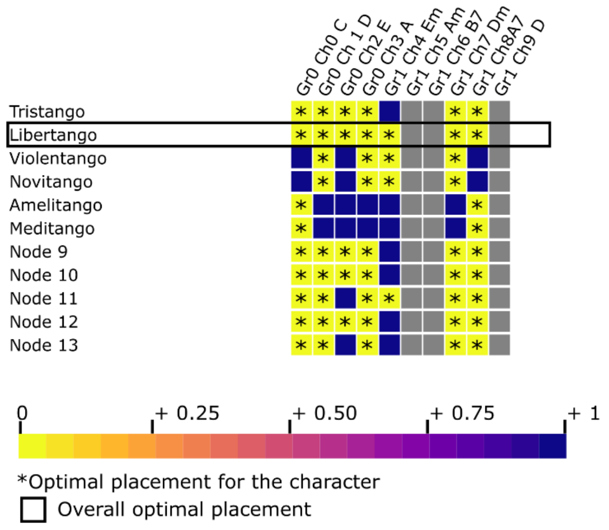
Query_MultOpt.svg

The output file shows the results of the multiple optimality criterion analysis. As in the previous case, color code in this case only represent the optimality in a relative way, because the difference in score in absolute values depend on the condition used to run each iteration.



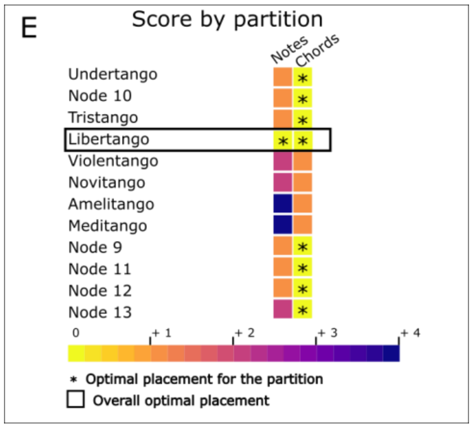
Chars_score_table.svg

An *svg* file including a colored chart showing the scores of each character when the target species is placed on each branch of the tree. In cases where multiple optimal positions are retrieved for the target species, one of these positions is considered as reference for character score calculation. The same information is included in a text file (*Chars_score_table.out*). The best placement(s) for each character is indicated with an asterisk and the optimal placement for the target species for the complete dataset is marked with a black rectangle.



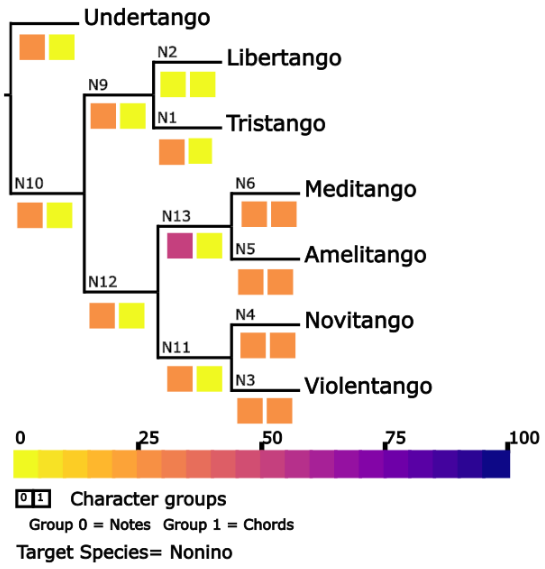
Partitions_score_table.svg

Similar to *chars_score_table.svg* but presenting the scores for each partition. These groups should be defined beforehand in the datamatrix using the command *xgroup*. The same information but in text format is included in the file *Partitions_score_table.out*. The best placement(s) for each partition is indicated with an asterisk and the optimal placement for the target species for the complete dataset is marked with a black rectangle.



Error_by_subtree_tree.svg

An *svg* file displays the "in subtree error" (*ISE*) for each subtree for each character partition. The errors are displayed as a colored checkerboard, with each square representing a different character partition.

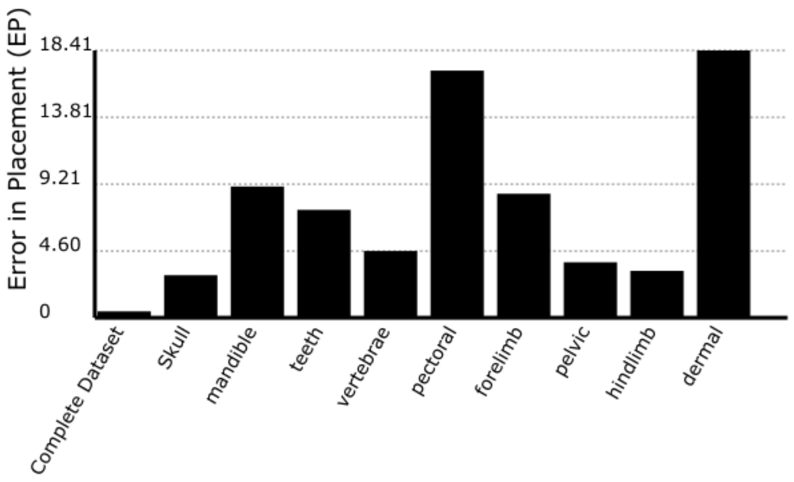


Error_by_subtree_table.out

A text file including the *ISE* and *OSE* values for each subtree for the complete dataset and each character partition. The file also included the number of species considered to calculate the error at each subtree.

Error_by_partition_barplot.svg

An *svg* file including a barplot that compares *EP* values for the complete dataset and for each character partition. Generated only when character partitions are defined.



Error_by_partition_table.out

This txt file includes the same information presented in the file "***Error_by_partition_barplot.svg"***. In addition, it includes the raw data considered to calculate this error (the error for each backbone species) on each partition and the information about the character and taxon sampling considered to calculate the error in each partition.

Opt_trees.tre, Opt_trees_N.tre, Opt_treesML.tre, Opt_treesMP.tre, Opt_treesIW.tre

Trees files in parenthetical notation including every optimal tree obtained when running the constrained analysis. When the sensitivity analysis is performed, one tree is generated for each condition.

ANALYSIS AVAILABLE UNDER THE DIFFERENT ANALYTICAL CONDITIONS

The script allows placing the query species on the backbone tree under all analytical conditions. The rest of the analyses are available for some analytical conditions. This is because some analyses: (i) cannot be run on TNT, (ii) are very time consuming or, (iii) are not conceptually sound. When the option "***Allow query species to modify relationships"*** is selected, no further analysis can be run, only placing a single target species on the tree. The same happens when dealing with multiple reference trees. Tables 1 and 2 show which analyses can be performed when a single fixed reference tree is considered.

	Sensitivity/ Leave-one-out validation analysis	**Multiple Optimality Criteria analysis**	**Error analysis**	**Character analysis**
No query species	No	No	Yes	No
Single query species	Yes	Yes	Yes	Yes
Multiple query species *	Yes	Yes	Yes	No

* Not available for Maximum likelihood

Table 1. Analysis that can be run under different selection of query/target species considering a single (and fixed) reference tree.

	Sensitivity Analysis	**Leave-one-out validation analysis**	**Multiple Optimality Criteria analysis**	**Error analysis**	**Character analysis**
Maximum Likelihood	Yes	No	Yes	No	Yes
Implied Weighting	Yes	Yes	Yes	Yes	Yes
Maximum Parsimony	Yes	Yes	Yes	Yes	Yes

Table 2. Analysis that can be run under different optimality criterion considering a single (and fixed) reference tree.

REFERENCES

Berger, S. A., Krompass, D., & Stamatakis, A. (2011). Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. Systematic biology, 60(3), 291-302.

Catalano SA, Escapa I, Pugh K., Hammond, A. Goloboff PA & Almécija S (2024). PlaceMyFossils: An integrative approach to analyze and visualize the phylogenetic placement of fossils using scaffolds.

Goloboff, P.A. (1993) Estimating character weights during tree search. Cladistics, 9(1), 83-91.

Goloboff, P.A. (2014). Extended implied weighting. Cladistics, 30(3), 260-272.

Goloboff, P.A., & Catalano, S. A. (2011). Phylogenetic morphometrics (II): algorithms for landmark optimization. Cladistics, 27(1):42–51.

Goloboff, P. A., & Morales, M. E. (2023). TNT version 1.6, with a graphical interface for MacOS and Linux, including new routines in parallel. Cladistics, 39(2), 144-153.

Goloboff, P., Mattoni, C., & Quinteros, A. (2006). Continuous characters analyzed as such. Cladistics, 22(6), 589–601.

Goloboff, P.A., Torres, A. and Arias, J.S., 2018. Weighted parsimony outperforms other methods of phylogenetic inference under models appropriate for morphology. Cladistics 34, 407– 437.

Klopfstein, S., & Spasojevic, T. (2019). Illustrating phylogenetic placement of fossils using RoguePlots: an example from ichneumonid parasitoid wasps (Hymenoptera, Ichneumonidae) and an extensive morphological matrix. PLoS One, 14(4), e0212942.

Torres, A., Goloboff, P. A., & Catalano, S. A. (2022). Parsimony analysis of phylogenomic datasets (I): scripts and guidelines for using TNT (Tree Analysis using New Technology). Cladistics, 38(1), 103-125.