Installation

For installing TreSpEx decompress the file and move the folders to any location you want. TreSpEx will have to be started from this location.

If pre-compiled databases shall be used for the blast searches these packages have to be downloaded at http://annelida.de/research/bioinformatics/software.html as it is rather larger and hence the download would be much slower. These packages have to be decompressed into the folder blast/bin/Databases within the TreSpEx folder. This will generate folders beginning with PrecompiledDatabases within the Databases folder. ALL folders, each containing a precompiled database for the blast search, within the PrecompiledDatabases folders have to be moved up into the Databases folder, so that the Databases folder contains these folders and NOT the PrecompiledDatabases folders.

For the same reason the folder with example files is can be download separately at http://annelida.de/research/bioinformatics/software.html. This package has also to be decompressed, but the location does not matter in contrast to the database package.

To execute TreSpEx a PERL interpreter must be installed on the current run system. Linux and Mac systems do normally not need a subsequent installation because the interpreter is a standard tool included in that system in advance, but the installation of the following additional perl packages is necessary:

File::Copy::Recursive
File::Copy
Statistics::LineFit
Statistics::Test::WilcoxonRankSum
Statistics::Zed 0.072
Cwd

The packages can be easily installed on Linux like UBUNTU v12.04 or higher and Mac operating systems by typing either. . .

sudo apt-get install File-Copy-Recursive <enter>
sudo apt-get install File-Copy-perl <enter>
sudo apt-get install Statistics-LineFit-perl <enter>
sudo apt-get install Statistics-Test-WilcoxonRankSum-perl <enter>
sudo apt-get install Statistics-Zed <enter>

. . . or ...

sudo cpanm File::Copy::Recursive <enter>
sudo cpanm File::Copy <enter>
sudo cpanm Statistics::LineFit <enter>
sudo cpanm Statistics::Test::WilcoxonRankSum <enter>
sudo cpanm Statistics::Zed <enter>
sudo cpanm Cwd <enter>

Using the sudo command will prompt the input of a password for a user with administrative rights.

To use cpanm you have to install app::cpanminus. Detailed instructions are available at search.cpan.org/~miyagawa/App-cpanminus-1.7001/lib/App/cpanminus.pm#__top . If you have administrative right you can install cpanminus for the system using the following command line in the terminal:
curl -L http://cpanmin.us | perl - --sudo App::cpanminus <enter>

For Windows operatin systems a PERL interpreter has to be installed ex post. I would recommend the ActivePerl interpreter, which can be downloaded for free under:

http://activeperl.softonic.de/

The additional PERL packages can be installed via the ActivePerl package manager.

For using TreSpEx open the terminal of your operating system. Move through your directory path to the folder where TreSpEx is placed.

Example files

The folder TreSpEx_Example contains example input files, which can be used to test the different functions of TreSpEx. The input files are taken from analyses of Struck [1] and Struck et al. [2]. Besides the input files, the results generated with these data are also included in the folder. In this folder are three subfolders PABA, Paralogy as well as Saturation_and_such, in which the results and input files of the different functions have been grouped as well as the relevant results from ML analyses. The structure of the examples folder is as follows:

TreSpEx_Example
- PABA
  - Combination_and_permutation
  - PABA_WSRT_and_Permutation
  - RAxML
  - Summary_bootstrap
- Paralogy
  - A_posteriori
  - A_priori
  - BLAST
  - Pruning
- Miscellaneous
  - Average_bootstrap
  - LBscore_TipToRoot_EvolDist
  - RAxML
  - Saturation

Usage

The chapter only provides a general outline of the usage of the different functions of TreSpEx. Details about the individual option and input files can be found in the section Options and Input file formats, respectively. TreSpEx contains 10 different functions. However, these functions can be grouped into three different types of analyses. The first type is the detection of paralogous sequences based on single-gene analyses as suggested by different authors [1, 3-5] and combines the functions a-d. The second one is the detection of conflict in datasets based on the PABA principle [2, 6] and combines the functions h-j. And the third is tree-based measurements of different forms of biases such as saturation or long branches [7-9] as well as the calculation of average bootstrap support for a tree [10] and combines the functions e-g.

General options

For all functions a path to the folder containing the files for the analysis can be set using the option –path *path-from-root*. (e.g., -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Paralogy/A_priori/). Alternatively, all files can be copied into the TreSpEx folder and the program used without the path option.

Help can be called using the option ***–h*** (see below).

Moreover, log files will be generated for the different functions.

Detection and pruning of paralogous sequences

The procedure of detection of paralogous sequences implemented in TreSpEx is an *a posteriori* procedure [1, 3-5]. This means, that by some way (e.g., HaMStR [11] or OrthoMCL, ReMark, MultiMSOAR 2.0 [12-14]) sets of supposedly orthologous genes have already been generated. However, as these processes have a certain chance to group paralogous genes together, which eventually might mislead the analysis of the combined data as for these sets the gene tree is inferred instead of the species tree [1, 3-5]. Therefore, the individual sets of supposedly orthologous sequences should be screened for paralogous sequences before the data are used for further analyses. To do so, single gene trees are reconstructed and screened for clades with strong bootstrap support, as clades reflecting the splitting between two sets of paralogous sequences within a gene tree will gain strong support. However, as strong nodal can also be related to actual support for natural grouping additional filtering steps are necessary and have been implemented in TreSpEx. For more detail on this method please refer to the corresponding literature [1, 3-5].

A priori screening (fun –a) / Example in folder Paralogy/A_priori

Command line:
perl TreSpEx.v1.pl ***-fun*** *a* ***-gts*** *Yor N* ***-lowbs*** *value* ***-upbs*** *value* ***-possc*** *0,1,2,3* ***-poslb*** *0,1,2,3* ***-lowbl*** *value* ***-upbl*** *value* ***-possb*** *0,1,2,3* ***-maxtaxa*** *value* ***-blt*** *value* ***-ipt*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun a -gts Y -lowbs 95 -upbs 100 -possc 1 -poslb 2 -lowbl 4 -upbl 4 -possb 3 -maxtaxa 3 -blt 0.00001 -ipt Input_bipartitions_trees.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Paralogy/A_priori/

This screening is done prior to any concatenation and analysis of the data. To use this function a list of names of tree files in Newick format with bootstrap support values has to be provided via ***–ipt*** *Name-of-file*. (e.g., -ipt Input_bipartitions_trees.txt). The tree files have to be in the same folder as the file containing the list. All trees in this list (i.e. the RAxML_bipartitions.-files) will then be screened for bootstrap support values.

All clades, which possess a certain BS value (e.g., 95 or 95 to 100), will be extracted from these trees for further analysis. Via the option ***–lowbs*** *value* and ***–upbs*** *value* the lower and upper bound of the range of bootstrap values will be defined. In the provided example –lowbs 95 and –upbs 100 have been used, so all clades with a bootstrap value from 95 to 100 will be extracted. If –lowbs 95 and –upbs 95 is chosen, all clades with a bootstrap value of 95 will be extracted.

These detected clades will be written to the file PotentialParalogsBootstrap.txt and the tab-delineated file NOT_filtered_Hits_per_BS_value.csv summarizes how many clades across all trees have been found at each bootstrap value.

The next steps are that certain detected clades can be masked for the further analyses and/or the remaining detected clades be sorted based on certain criteria.

Masking of detected clades can be invoked to exclude clades from further analyses, for which there is evidence that the clade is a natural unit. For example, if a clade consists only of mammalian species it is more likely that the strong bootstrap is due to actual signal rather than being an artifact. To mask clades the option ***–gts*** has to be used (e.g., -gts Y). Moreover, a file or files beginning with ***GroupedTaxa*** and ending with ***.txt*** (e.g., GroupedTaxaFamily.txt) has to be provided. In this file each line defines a clade, which shall be masked. More than one file can be provided if different settings shall be tested. For example, masking clades based on different taxonomic levels (e.g, belonging to the same genus, class or phylum) can be done in that way. TreSpEx will use each file separately and write the results into individual folders. Therefore, the text between GroupedTaxa and .txt will be taken as a specifier and the folder named correspondingly beginning with TaxaScreening (e.g., TaxaScreeningFamily). TreSpEx will mask all clades, which consists only of taxa provided by the individual lines. For example, the file GroupedTaxaFamily.txt contains in the first line the names of all clitellate taxa present across all trees, even if not all clitellate species are present in all individual trees. For each individual tree all clades, which consists only of clitellate species as defined by the first line in the file GroupedTaxaFamily.txt will then be masked for further analyses.

The detected clades, which are not masked, will be saved in the file Pruned_PotentialParalogsBootstrap.txt in the corresponding result folder. Moreover, the corresponding trees, which contain a detected clade listed in Pruned_PotentialParalogsBootstrap.txt, will be compiled into the single file PrunedTrees_PotentialParalogsBootstrap.tre.

Besides masking the detected clades can be sorted based on different criteria. The criteria are strong conflict (invoked by ***–possc***), long branch (invoked by ***–poslb***) and short branch (invoked by ***–possb***). This sorting is done in an ordered fashion provided by the user. In the example, the criterion strong conflict is checked first (i.e., -possc 1). If the clade matches this criterion it is written to the result files of this criterion. If not, the criterion long branch is checked next (i.e., -poslb 2). If that does not fit, the criterion short branch is checked (-possb 3). If that also does not fit, the clade is sorted into a result file compiling the remaining files. The user can invoke any combination of these criteria. For example, -possc 0 –poslb 1 –possb 2 would have the order long branch followed by short branch and the criterion strong conflict would not be invoked at all.

As for the masking the clades written into a corresponding result file (e.g., Pruned_PotentialStrongConflictBootstrap.txt) and the tree are compiled into one file (e.g., PrunedTrees_PotentialLongBranchParalogsBootstrap.tre) as well. Moreover, the tab-delineated file PrunedHits_per_BS_value.csv will be generated summarizing how many non-masked clades across all trees have been found at each bootstrap value for each criterion.

The *criterion strong conflict* means that the detected clade is in conflict with an *a priori* defined clade for masking. Therefore, to use the strong conflict criterion the masking procedure is a prerequisite to first filter out all clades, which are in agreement with this clade. Thus, all clades left must be a composition of members of at least two different groups of taxa (e.g., flatworms and an annelid). Invoking the strong conflict criterion now checks if a member of the predefined clade in GroupedTaxa*.txt file from above, which was not present in the detected clade, is present in the tree some where else. If that is the case, a conflict occurred between a clade with evidence for monophyly and a strongly supported clade found in the corresponding tree. For example, the file GroupedTaxaFamily.txt contains in the first line the names of all clitellate taxa as Clitellata is well established as a monophyly at present. If now a tree has a strongly supported clade of some clitellates and a polychaete, while other clitellates are present in the tree, but not in this clade, this finding contradicts monophyly of Clitellata and hence is at conflict with it.

The *criterion long branch* checks if a long internal branch is leading to the detected clade. TreSpEx determines if the internal branch leading to the detected clade is at least x times longer than the average of all internal branches in the tree. The x value has to be provided by the user using ***–lowbl*** *value* and ***–upbl*** *value*. For example, using –lowbl 4 and –upbl 4 the branch has to be at least 4 times longer. If the results based on different minimal x values shall be investigated this can be done in a single run by, for example, –lowbl 4 and –upbl 6. Then it will be tested if the branch is at least 4 times, 5 times or 6 times longer. The results of each x value will be written into an individual folder named LBfactor plus the x value (e.g., LBfactor4). This means, that at least 5 times longer is a subset of the results of at least 4 times longer, and at least 6 times of both them.

The *criterion short branch* checks if at least one of taxa in the detected clade has a terminal branch length below a certain threshold. This criterion can be used to detect possible contamination issues between taxa of the dataset by looking for zero- or close to zero-branch length taxa. As it is seems that, for example, RAxML does calculate true zero terminal branch length (although they appear that way in the figure, in the tree file they still possess a very short branch) a threshold has to be provided by ***-blt*** *value* (e.g., -blt 0.00001). All terminal branch length below this value will be regarded as short-branched. However, as in very conserved genes most terminal branches have nearly zero branch length this is cannot be taken as an indication of contamination in such cases. As it is very unlikely that contamination occurred between all or most taxa of a dataset a maximum number of taxa present in a detected clade has to be provided by ***–maxtaxa*** *value* (e.g., -maxtaxa 3). If more taxa are present in clade, TreSpEx will not check for short branches.

A posteriori screening (fun –b) / Example in folder Paralogy/A_posteriori

Command line:
perl TreSpEx.v1.pl ***-fun*** *b* ***-gts*** *Yor N* ***-lowbs*** *value* ***-upbs*** *value* ***-possc*** *0,1,2,3* ***-poslb*** *0,1,2,3* ***-lowbl*** *value* ***-upbl*** *value* ***-possb*** *0,1,2,3* ***-maxtaxa*** *value* ***-blt*** *value* ***-ipt*** *Name-of-file* ***-trf*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun b -gts Y -lowbs 95 -upbs 100 -possc 1 -poslb 2 -lowbl 4 -upbl 4 -possb 3 -maxtaxa 3 -blt 0.00001 -ipt Input_bipartitions_trees.txt -trf Tree_Fig3.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Paralogy/A_posteriori/

In contrast to the *a priori* screening (-fun a), herein the screening is done only for clades provided via a tree file. All detected clades not congruent with these clades will be masked. By this way one can *a posteriori* test if clades present, for example, in the best tree of an analysis of the concatenated data are potentially due to the misleading potential of paralogy. The tree file has not be completely resolved and is provided via the option ***–trf*** *Name-of-file* (e.g., -trf Tree_Fig3.txt). All resolved clades in that tree will be determined and saved in the file GroupedTaxa_A_posteriori_Groups.txt. All detected clades not in accordance with these clades will be masked and the remaining detected clades be written to the file AP_PotentialParalogsBootstrap.txt for further analyses.

Otherwise, it is the same as -fun a and the same options can be used.

Blast search against reference database (fun –c) / Example in folder Paralogy/BLAST

Command line:
perl TreSpEx.v1.pl ***-fun*** *c* ***-ppf*** *Name-of-file* ***-ipt*** *Name-of-file* ***-ipa*** *Name-of-file* ***–db1*** *Name-of-database* ***–db2*** *Name-of-database* ***-ediff*** *value* ***-ltp*** *value* ***-utp*** *value* ***-evalue*** *value* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun c -ppf Pruned_PotentialParalogsBootstrap.txt -ipt Input_bipartitions_trees.txt -ipa Input_alignments.txt -db1 Bos_taurus -db2 Branchiostoma_floridae -ediff 5 -ltp 0.0 -utp 1.0 -evalue 1e-20 -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Paralogy/BLAST/

To determine if suspicious clades detected by the screening procedures above or any other one are indeed paralogous sequences blast searches of sequences of that alignment against two reference databases can be conducted in TreSpEx. This allows comparing if the same hits are returned for the suspicious sequences and the other sequences in the alignment. Hereby, TreSpEx regards the smaller of the two groups of the bipartition as the one with the suspicious sequences. TreSpEx will sort the results of the blast searches into four different categories given certain threshold condition are met (see below): no hits in either all suspicious sequences, all other sequences or all sequences, certain paralogy, no paralogy and uncertain cases.

Therefore, a list of the suspicious clades, which shall be tested for paralogy by blast searches, has to be provided via the option ***–ppf*** *Name-of-file* (e.g., -ppf Pruned_PotentialParalogsBootstrap.txt). The file has to have the same format as the result files generated by the two above screening procedures (e.g., Pruned_PotentialParalogsBootstrap.txt). Each new line defines a clade and starts with the name of the tree file followed by the bootstrap value and the branch length of that clade and then by the names of taxa of the clade as they are in the tree and the corresponding alignment.

To be able to relate the name of the tree in each line to an alignment a list of names of tree files has to be provided via the option ***-ipt*** *Name-of-file* (e.g., -ipt Input_bipartitions_trees.txt), which is in the same order as a list of corresponding relaxed phylip files provided via the option ***-ipa*** *Name-of-file* (e.g., -ipa Input_alignments.txt). Both the tree files and the alignment files have also to be provided as some cross-checks are done between these two files and the taxa names of the clade.

Two reference databases, against which the blast search is conducted, have to be chosen via the options ***-db1*** *Name-of-database* and ***-db2*** *Name-of-database* (e.g., -db1 Bos_taurus -db2 Branchiostoma_floridae). Ideally, these databases are non-redundant, that is each orthologous sequence is represented in the database only by one sequence. Pre-compiled databases (http://annelida.de/research/bioinformatics/software.html) can also be used if placed in the folder blast/bin/Databases (see Installation). However, own databases can be deposited in the same folder as fasta files. TreSpEx will convert them into databases for blast searches.

The threshold of the e value can be set for the blast searches using the option ***–evalue*** *value* (e.g., -evalue 1e-20). TreSpEx will automatically determine, which blast search has to be conducted give the database and the alignment file. The results of the blast search

of both the suspicious sequences and the other sequences will be placed in the folder blast_results, which will contain the text files with the actual blast results, the alignment files used and a log file.

Based on the results of the blast searches against the two reference data the results for each suspicious clade will be sorted into different categories. First, it is determined if at least one of the blast searches returned hits for both suspicious and other sequences. If that is not given the corresponding clade information will be written to the file No_hits_Pruned_PotentialParalogsBootstrap.txt in the folder blast_comparisons/No_hits. All remaining detected clades will be classified as certain paralogy, no paralogy and uncertain cases. Therefore, for each suspicious sequence the proportion of best hits in the searches of the other non-suspicious sequences is determined, which is identical with the best hit of that suspicious sequence. For the sorting two threshold values have to be provided via ***-ltp*** *value* and ***-utp*** *value* (e.g., -ltp 0.0 -utp 1.0). The ltp value provides a lower threshold. If for ALL suspicious sequences in the searches against one of the two databases this proportion is equal to or lower than this lower threshold this detected clade is regarded as a certain case of paralogy and written to the file Certain_Pruned_PotentialParalogsBootstrap.txt in the folder blast_comparisons/ Certain_paralogy. For example with -ltp 0.0, if the clade of suspicious sequences contains three sequences and for all three sequences the proportion of identical best hits is 0 in the search against either database 1 or 2 this case is given. On the other hand, the utp value provides an upper threshold. If for ALL suspicious sequences in the searches against both databases this proportion is equal to or higher than this upper threshold this detected clade is regarded as a certain case of none paralogy and written to the file No_Pruned_PotentialParalogsBootstrap.txt in the folder blast_comparisons/No_paralogy. For example with -utp 1.0, if the clade of suspicious sequences contains three sequences and for all three sequences the proportion of identical best hits is 1.0 in the searches against both databases 1 and 2 this case is given. All remaining clades will be written to the file Uncertain_Pruned_PotentialParalogsBootstrap.txt in the folder blast_comparisons/ Uncertain_paralogy.

Besides the files with clade information mentioned above also a file is generated in the corresponding folder with the proportion of identical sequences for each suspicious sequence in the searches against both databases, for further exploration by the user if needed. Moreover, these file contain not only the proportion of identical best hits, but also the proportion of identical hits within a confidence set of hits. Therefore, the option ***–ediff*** *value* (e.g., -ediff 5) has to be provided. This value provides the difference in the order of magnitude by which the e value can be higher to be still considered part of the confidence set of best hits. For example, a value of 5 and a best hit with an e value of e^{-78} means that all hits will be considered for this set, which have an e value not higher than e^{-73}. However, this is only provided for further exploration by the user and is not relevant for the sorting by TreSpEx.

Finally, the results of the blast searches against both datasets comprising only the best hits and the hits of the confidence set are compiled for each clade of suspicios sequences into a single file (e.g., Compiled_results_23816Ali_22.txt) within the folder blast_comparisons.

Pruning of suspicious sequences from the individual datasets (fun –d) / Example in folder Paralogy/Pruning

Command line:
perl TreSpEx.v1.pl ***-fun*** *d* ***-ppf*** *Name-of-file* ***-ipt*** *Name-of-file* ***-ipa*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun d -ppf Certain_Pruned_PotentialParalogsBootstrap.txt -ipt Input_bipartitions_trees.txt -ipa Input_alignments.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Paralogy/Pruning/

Using this function the sequences of a suspicious clade can be pruned from the corresponding relaxed phylip files. Therefore, a list of the suspicious clades, which shall be pruned, has to be provided via the option ***–ppf*** *Name-of-file* (e.g., -ppf Certain_Pruned_PotentialParalogsBootstrap.txt). The file has to have the same format as the result files generated by the two above screening procedures or by the blast searches in function c (e.g., Pruned_PotentialParalogsBootstrap.txt or Certain_Pruned_PotentialParalogsBootstrap.txt). Each new line defines a clade and starts with the name of the tree file followed by the bootstrap value and the branch length of that clade and then by the names of taxa of the clade as they are in the tree and the corresponding alignment. Hereby, TreSpEx regards the smaller of the two groups of the bipartition as the one with the suspicious sequences to minimize the number of sequences to be pruned from the dataset. As the sequences of the suspicious clade are paralogs of the other clade of the bipartition this is also true the other way around, the sequences of the sequences of the other clade of the bipartition are the paralogs of the suspicious clade. The information for the smaller clade will be written to the file IngroupOutgroup_*Name-of-file* (e.g., IngroupOutgroup_Certain_Pruned_Potential ParalogsBootstrap.txt)

To be able to relate the name of the tree in each line to an alignment a list of names of tree files has to be provided via the option ***-ipt*** *Name-of-file* (e.g., -ipt Input_bipartitions_trees.txt), which is in the same order as a list of corresponding relaxed phylip files provided via the option ***-ipa*** *Name-of-file* (e.g., -ipa Input_alignments.txt). Both the tree files and the alignment files have also to be provided as some cross-checks are done between these two files and the taxa names of the clade.

As there can be more than two sets of suspicious sequences within an individual dataset a first step of TreSpEx is to compile all instances of suspicious clades into a single line for each individual dataset and write this information to the file Compiled_Certain_Pruned_PotentialParalogsBootstrap.txt. Then the sequences will be pruned from the corresponding relaxed phylip alignment file and the modified alignment saved in the folder FilesPruned. All alignment files from which no sequences were pruned will be copied into the folder FilesNotPruned.

Detection of conflict

TreSpEx in combination with a program for phylogenetic reconstruction such as RAxML [15] or PhyloBayes [16] can also be used to detect conflicts in datasets based on the PABA principle [2, 6]. Using this principle conflict is detected on a node-by-node and partition-by-partition basis utilizing nodal support values. The PABA (Partition Addition Bootstrap Alteration) principle was first proposed using bootstrap values [2], but can also be conducted with any other nodal support values such as Bremer support (PABSA) or posterior probabilities (PAPPA) [6]. For reasons of simplicity it will be referred to in the following as PABA. To conduct a PABA analyses four steps have to be done.

First, the partitions of the datasets have to be combined in all possibilities or in a range of possibilities (function h). The latter means, that, for example, if a dataset has 10 partitions the user can also choose to generate only all possible combination with, for example, 8 or 9 combined partitions. Second, a phylogenetic reconstruction of each of these combinations has to be conducted using a program of the users choice. Third, the nodal support values for each bipartition have to be summarized across all datasets of the second step. If the program used by the user provides for each dataset a file, in which all tree files for determining the support are compiled, the function i of TreSpEx can be used to for this summary step. For example, RAxML provides all trees of the bootstrap analysis in a files beginning with RAxML_bootstrap*, while PhyloBayes compiles all trees of the chain in the file *.treelist. Fourth, the actual PABA analysis has to be conducted, which determines how the support for a bipartition (node) changes as a partition is added given the order of addition (as 8^{th} or 9^{th} partition added for example). This can be conducted using function j in TreSpEx.

Moreover, using TreSpEx two tests of significance of the PABA results can be conducted as proposed by Struck [6] derived from a test developed for ILD tests [17, 18] and from the Tempelton test [19-23]. First, using a permutation test it can be assessed for each node and partition if the same PABA value can be obtained just by chance due to randomly partitioning the data set in partitions of same sizes as defined partitions. To conduct the permutation test all four steps of the PABA analysis above (including the usually time consuming phylogenetic reconstruction) have to be done not only for the original dataset, but also for each of the, for example, 100 permutations of the original dataset. The second test employs a Wilcoxon-Signed-Rank test (WSRT) to determine if for a given set of bipartition (e.g., all nodes of the best tree or an alternative tree) the addition of a partition is overall beneficial to nodal support or detrimental. Thus, it can be assessed if the addition of partition its positive contribution outweighs, if present, a negative impact on a just few nodes or not. Hence, it can guide the decision if a partition should be completely excluded.

Combination of partitions and generation of corresponding datasets (fun –h) / Example in folder PABA/Combination_and_permutation

Command line:
perl TreSpEx.v1.pl ***-fun*** *h* ***-aln*** *Name-of-file* ***-partf*** *Name-of-file* ***-ubnp*** *value* ***-lbnp*** *value* ***-per*** *Y/N* ***-nper*** *value* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun h -aln Eunicida4Genes_nonInterleaved.phy -partf Partitions_Eunicida.txt -ubnp 4 -lbnp 1 -per Y -nper 100 -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/PABA/

Using this function the partitions of a dataset will be combined in all possibilities for a given range of numbers of partition to be combined. This range will be determined by the options ***-ubnp*** *value* and ***-lbnp*** *value* (e.g., -ubnp 4 -lbnp 1). For example, if the dataset consists of 4 partitions -ubnp 4 -lbnp 1 would mean that all possible combinations of partitions will be generated ranging from combining only 1 to 2, 3 and 4. However, if the dataset comprised 5 or more partitions not all possible combinations would be generated, but only the ones in which up to four partitions are combined. Within this range though all possible generations will be generated.

The alignment of the dataset is provided via the option ***–aln*** *Name-of-file* (e.g., -aln Eunicida4Genes_nonInterleaved.phy) in the relaxed phylip format and a definition of partition by the option ***-partf*** *Name-of-file* (e.g., -partf Partitions_Eunicida.txt). The format of the latter is similar to partition definition used by RAxML. Finally, if a permutation test shall be conducted permutated datasets will generated (including the same combinations of partitions as for the original dataset) the option ***–per*** has to be set to Y or Yes and the number of permutations to be conducted via the option ***–nper*** *value* (e.g., -nper 100).

TreSpEx will write out relaxed phylip files for each generated combination into the folder Combinations. In parallel is will as generate a corresponding partition file, if partitioned phylogenetic analyses of these data shall be conducted. If permutated data are generated as well, a corresponding permutated relaxed phylip file for each combination will be written to an individual subfolder for each permutation. For these permutated files the same partition file applies as for the original dataset. For the sake of reproducibility each permutation of the sequence positions is compiled in the file Permutations_used.txt.

Summarizing nodal support across different datasets (fun –i) / Example in folder PABA/Summary_bootstrap

Command line:
perl TreSpEx.v1.pl ***-fun*** *i* ***-tf*** *Name-of-file* ***-mint*** *value* ***-ipt*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun i -tf TaxaNames.txt -mint 1 -ipt Bootstrap_trees_Original.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/PABA/RAxML/Original/

Using this function the bipartition table for each compilation file, in which all tree files for determining the support for the analyses of a dataset are compiled (e.g., RAxML_bootstrap* files of RAxML or *.treelist of PhyloBayes), is generated and these are summarized into one matrix for all files. A plain text file, which lists the names of all such compilation files to be summarized, is provided via the option ***-ipt*** *Name-of-file* (e.g., -ipt Bootstrap_trees_Original.txt). For each bipartition present in any of the trees in that compilation file the number of occurrences is counted and the support value is calculated as a percentage of all trees present in that compilation file. These are written into a bipartition table for each file (i.e., dataset) if the support value of the bipartition is equal to or larger than a threshold value. The file name begins with Bipartition_Table_*. Via option ***–mint*** *value* (e.g., -mint 1) this threshold is provided.

Then the results of these individual bipartition tables are compiled into one matrix for all files (i.e., dataset) analyzed in this run. If a bipartition is present in the tables of some datasets, but not in the others the threshold value minus 1 is written as lowest possible value for all instances, which lack this bipartition. For example, if the threshold is 1, this value would be 0 for all datasets, which would not have this bipartition. To generate the summary table it is necessary that all trees have the same set of taxa and this is checked using a list of taxon names via the option ***-tf*** *Name-of-file* (e.g., -tf TaxaNames.txt). This prerequisite is necessary as it is otherwise not possible to align the bipartitions across different datasets to each other. The summary matrix is written to the file Compiled_Bipartition_Tables.txt.

If a permutation test shall be conducted in function j this summary has to be done for each permutation.

PABA analyses and significance tests (fun –j) / Example in folder PABA/PABA_WSRT_and_Permutation

Command line:
perl TreSpEx.v1.pl ***-fun*** *j* ***-per*** *Y/N* ***-wsrt*** *Y/N* ***-partf*** *Name-of-file* ***-tabf*** *Name-of-file* ***–mint*** *value* ***-maxt*** *value* ***-ori*** *Name-of-file* ***-bpf*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun j -per Y -wsrt Y -partf Partitions_Eunicida.txt -tabf Tabulated_bootstrap_files.txt -mint 0 -maxt 100 -ori Original_Compiled_Bipartition_Tables.txt -bpf Biparts_for_WSRT.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/PABA

Using this function the actual PABA analysis is conducted. Therefore, a definition of partition has to be provided by the option ***-partf*** *Name-of-file* (e.g., -partf Partitions_Eunicida.txt), which has the format similar to the partition definition used by RAxML. Moreover, via ***-tabf*** *Name-of-file* (e.g., -tabf Tabulated_bootstrap_files.txt) a list of names of tabulated summary files of support values is provided. These summary files (e.g., Original_Compiled_Bipartition_Tables.txt) have to have a format similar to the one generated by –fun i (see Input file formats). For all summary files of this list a PABA analysis conducted. The results of each analysis is written into a file beginning PABA_results_woSignificance_* with in separate Results folder within the folder PABA_calculations.

As some measurements of nodal support such as bootstrap values or posterior probabilities have upper and lower bounds (e.g., bootstrap values can only be as high as 100 or as low as 0) the PABA calculations would be misled in cases where these upper and lower values are not altered by the addition of a partition [2, 6]. Therefore, such cases will not be further considered for the PABA calculation. To do so, the user has to provide the upper and lower bounds in the summary files using the options ***–mint*** *value* and ***-maxt*** *value* (e.g., -mint 0 -maxt 100). These values have not always to be 0 and 100. For example, posterior probabilities –maxt is more likely to be 1.0. If the summary files were generated with the function i of TreSpEx and –mint was set to, for example, 5 then (as it is, for example, the default setting for bipartition tables in PAUP), then using this function –mint would have to be 4 (i.e., 5-1; see function i) as this would be lowest value possible in the summary files generated by function i.

If a WSRT shall be conducted the option ***–wsrt*** has to set to Y or Yes (e.g., -wsrt Y). Moreover, via option ***-bpf*** *Name-of-file* (e.g., -bpf Biparts_for_WSRT.txt) a list of names of files has to be provided, each of which contains a set of bipartitions to be tested by the WSRT. The format of these files is simple: each new line provides the identifier of the bipartition (e.g., iiiiiiiiiiiiioi or a specific name) and all identifier present in this file will be regarded as one set. Therefore, for each set, which shall be test its own file has to be generated (e.g., Nodes_28Sonly.txt and Nodes_All4Genes.txt). It is important to notice that the identifiers of the bipartitions have to be the exact same bipartition identifiers as used in the first column of the tabulated summary file. The results of the WSRT will be written into the file beginning with WSRT_results_ in the same folder as the results of the PABA calculation.

If a permutation test shall be conducted the option ***–per*** has to set to Y or Yes (e.g., -per Y). Moreover, via option ***-ori*** *Name-of-file* (e.g., -ori Original_Compiled_Bipartition_Tables.txt) the name of the summary file of the original dataset, which shall be tested,

has to be provided. Moreover, this name has also to be in the list of names provided by option ***–tabf***. This list has also to include the names of the summary files of all permutated datasets, but no other datasets as all summary files from this list except the original one will be taken for the permutation test and compared to the results from the original data. The results of the permutation test are written into the file beginning with PABA_results_Significance_ in the Results folder of the original data in the folder PABA_calculations.

Miscellaneous

Indices for long-branched taxa based patristic distances in a tree (fun –e) / Example in folder Miscellaneous/LBscore_TipToRoot_EvolDist

Command line:
perl TreSpEx.v1.pl ***-fun*** *e* ***-ipt*** *Name-of-file* ***-tf*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun e -ipt Input_bipartitions_trees.txt -tf TaxaNames.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Saturation_and_such/LBscore_TipToRoot_EvolDist

Using this function, different indices for long-branched taxa can be calculated based on patristic distances (PD), i.e. the distance between two taxa based on the connecting branches, in a tree. One index, which is often used as a proxy in this regard, is the evolutionary rate of a dataset [8, 9]. To gauge the evolutionary rate the average PD between any pair of taxa in the tree is calculated instead of an uncorrected genetic distance p calculated from the alignment. Another measurement is the tip-to-root distance of each taxon to detect long-branched taxa [8, 9], which is also calculated by TreSpEx. Finally, the new LB score is calculated. To assess the branch length heterogeneity in a tree the LB score is based on the mean pairwise PD of a taxon *i* to all other taxa in the tree relative to the average pairwise PD over all taxa (a):

$$LB_i = \left(\frac{\overline{PD_i}}{\overline{PD_a}} - 1 \right) * 100$$

.

In specific, the score measures for each taxon the percentage deviation from the average and is independent of the root of the tree in contrast to tip-to-root indices.

Therefore, a list of names of tree files in Newick format with branch length has to be provided via ***–ipt*** *Name-of-file*. (e.g., -ipt Input_bipartitions_trees.txt). For each tree file the calculations above will be performed. To generate the taxon vs tree matrices for the taxon-specific indices (i.e., tip-to-root distance and LB score) it is necessary to provide a list of taxon names, which occur across all trees analysed, via the option ***-tf*** *Name-of-file* (e.g., -tf TaxaNames.txt).

In addition to the taxon-specific values, for both tip-to-root distances and LB scores tree-specific indices will be calculated to allow a direct comparison of the results of different trees. These indices for each tree are for both indices the average of the upper quartile of values as well as the standard deviation of the values as an indication of heterogeneity in the data. These four indices and the average PD of each tree will be saved in the file

LB_scores_summary_perPartition.txt. The taxon vs. tree matrices will be written to the files LB_scores_perTaxon.txt and TR_scores_perTaxon.txt for and LB scores and tip-to-root distances. Moreover, the taxon vs. taxon PD matrices of each tree will be saved in the folder PD_matrices.

Index for saturation based patristic distances in a dataset (fun –g) / Example in folder Miscellaneous/Saturation

Command line:
perl TreSpEx.v1.pl ***-fun*** *g* ***-ipt*** *Name-of-file* ***-ipa*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun g -ipa Input_alignments.txt -ipt Input_bipartitions_trees.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Saturation_and_such/Saturation

Using this function, two indices for saturation in dataset have been proposed, which are derived from plotting the patristic distances (PD), i.e. the distance between two taxa based on the connecting branches in a tree, against the uncorrected genetic distance p calculated from the alignment for each pair of taxa [7]. One index, which has been used, is the slope of the linear regression of this plot as well as the R^2 fit of the data to this slope. The higher the slope or the better the fit the less saturated is the dataset.

Therefore, a list of names of tree files in Newick format with branch length has to be provided via ***–ipt*** *Name-of-file*. (e.g., -ipt Input_bipartitions_trees.txt), which is in the same order as a list of corresponding relaxed phylip files provided via the option ***-ipa*** *Name-of-file* (e.g., -ipa Input_alignments.txt). For each pair of tree and alignment file the indices above will be calculated.

The taxon vs. taxon matrices of both the PD and p distance of each tree and alignment will be saved in the folders PD_matrices and P_matrices, respectively. The slope and R^2 of each plot will be written to the file Correlation_Slope_Summary.txt.

Average bootstrap support of trees (fun –f) / Example in folder Miscellaneous/Average_bootstrap

Command line:
perl TreSpEx.v1.pl ***-fun*** *f* ***-ipt*** *Name-of-file* ***-path*** *Path-from-root*

Example command line:
perl TreSpEx.v1.pl -fun f -ipt Input_bipartitions_trees.txt -path /Users/torstenstruck/Documents/Data/Arbeit/Auswertung/TreSpEx_Example/Saturation_and_such/Average_bootstrap

Average bootstrap support has been used to assess if the alteration of certain conditions such as the addition of data or taxa was beneficial or detrimental for the phylogenetic reconstruction [24]. Therefore, a list of names of tree files in Newick format with bootstrap values has to be provided via ***–ipt*** *Name-of-file*. (e.g., -ipt Input_bipartitions_trees.txt). For each tree the average bootstrap support across all

resolved nodes in the tree will be calculated and written into the file Average_BS_perPartition.txt.

Input file formats

Tree files have to be in the Newick format as they are, for example, generated by RAxML (e.g., RAxML_besttree..., RAxML_bootstrap... or RAxML_bipartition....). For all analyses except the tip-to-root distance TreSpEx does not take the root of the tree into account. Thus, TreSpEx essentially works with unrooted trees even if rooted trees are provided.

Alignment files have to be in the relaxed sequential phylip format and database files in the fasta format.

Sequences being suspected to be paralogs to the other sequences in the alignment and which shall be blasted against reference databases or pruned from alignments will be provided via a plain text file with following format:
name-of-treefile bootstrap-support branch-length name-of-sequence name-of-sequence name-of-sequence…
For example,

RAxML_bipartitions.RPL24Ali.phy_fastBoot_Out.txt 96 1.75007055 Lanice_conchilega Eisenia_fetida
RAxML_bipartitions.22680Ali.phy_fastBoot_Out.txt 99 0.25451239 Pomatoceros_lamarckii Alvinella_pompejana
RAxML_bipartitions.22433Ali.phy_fastBoot_Out.txt 100 4.17620957 Scoloplos_armiger Sthenelais_boa Eurythoe_complanata_EST
RAxML_bipartitions.22606Ali.phy_fastBoot_Out.txt 100 0.99360951 Cerebratulus_lacteus Owenia_TS

Some options require the listing of taxon names, names of files containing lists of bipartition sets to be tested, the names of the tabulated summary files for the PABA calculation, names of tree files or names of alignment files via a plain text file. Be aware that in the latter two cases these names have to be in the same order, so that the name of a tree file at that position corresponds to the name of the alignment file, from which the tree is was reconstructed, at the same position in the other file. The format of these files is always the same. A new line provides each name:

	Taxa example	or tree example	or alignment example
Name-1	Eurythoe_complanata	RAxML_bipartitions.21904Ali.phy_fastBoot_Out.txt	21904Ali.phy
Name-2	Terebratalia_transversa	RAxML_bipartitions.21912Ali.phy_fastBoot_Out.txt	21912Ali.phy
Name-3	Arenicola_marina	RAxML_bipartitions.21942Ali.phy_fastBoot_Out.txt	21942Ali.phy
Name-4	Bugula_neritina	RAxML_bipartitions.21952Ali.phy_fastBoot_Out.txt	21952Ali.phy
…	…	…	…

Some options require the definition of the partitions. The format of this plain text file is similar to the format used by RAxML, though it is not required to provide a substitution model, but it can be done. Each new line provides a new definition, which has the following general format:
name-of-substitution-model, *name-of-partition* = *definition-of-partition*
The definition of the partition can have different possibilities. The simplest definition is that the partition starts at a position and ends at one (e.g., 1-500). However, it is also possible that the partition is located in two or more parts of the datasets (e.g., 250-350, 400-500, 700-800) or that codon positions are defined (e.g., 1-333\3 for first codon position, 2-333\3 for second and 3-333\3 for third). Finally, all these can be combined into a single file, so that a file can either look like this:

```
DNA, Partition1 = 1-600
WAG, Partition2 = 601-700
DNA, Partition3 = 701-900
...
```

or like this:

```
DNA, Partition1 = 1-300\3, 2-300\3
DNA, Partition2 = 3-300\3, 701-900
DNA, Partition3 = 301-400, 461-600
DNA, Partition4 = 401-460\3
DNA, Partition5 = 402-460\3
DNA, Partition6 = 403-460\3
WAG, Partition7 = 601-700
...
```

For the PABA calculation tabulated summary files of the support values are required in a format as is generated by –fun i. This is a plain text file in which the columns of the table are separated by tabs (this is important as white spaces are allowed to list taxa names of a bipartition with a single column). In this table the first row defines the names of the datasets, which have to include the names of partitions used to build the supermatrix and as they are defined in the also provided partition file (as it will be the case when the datasets have been generated with –fun h). For example taken the left example from above, if the dataset was build from Partition1 and Partition2 the name of the dataset has also to include these names, for example, as Partition1Partition2. The first column of the table serves as the identifier of the bipartition (e.g. iiiiiiiiiiiiioi or a specific name) and which will also be used in the files determining sets of bipartitions for WSRT tests. Then additional can be provided, but have to include at least one non-digit character (this requirement is not necessary for the first column). This can be used, for example, to provide proper taxon names for the each of the two bipartitions. After these columns the columns with the support values (e.g., bootstrap values) follow and each column has as a heading the name of the corresponding dataset, which includes the names of the partition present in this dataset. The general setup for three partitions in all combinations would be as follows:

		Part1	*Part2*	*Part3*	*Part1Part2*	*Part2Part3*	*Part1Part3*	*Part1Part2Part3*
id1	*t1 t2 t3*	*value*	*value*	*value*	*value*	*value*	*value*	*value*
id2	*t2 t4*	*value*	*value*	*value*	*value*	*value*	*value*	*value*
id3	*t5 t6*	*value*	*value*	*value*	*value*	*value*	*value*	*value*
id4	*t19 t1*	*value*	*value*	*value*	*value*	*value*	*value*	*value*
…	*…*	*…*	*…*	*…*	*…*	*…*	*…*	*…*

For an example see the files provided in the
TreSpEx_Example/PABA/PABA_WSRT_and_Permutation.

For the Wilcoxon-Signed-Rank test sets of bipartitions have to be provided. For the whole set it will tested if the contribution of a partition over all bipartitions in that set is more positive or negative based on the PABA results. The format of the plain text file is simple as each new line provides the identifier of the bipartition (e.g. iiiiiiiiiiiiioi or a specific name). Important: This identifier has to be the exact same bipartition identifier as used in the first column of the tabulated summary file of the support values across the different datasets.

	For example: Nodes_28Sonly.txt	or Nodes_All4Genes.txt
id1	iiiiiiiiiiiiioi	iiiiiiiiiiiiioi
id2	iioiiiiiiiiiooo	ooooooooooooioi
id3	ooioooooooooooi	iiiiiiiiiiiiooo
id4	iioiiiiiiiiiooo	ooooooiiooooooo
…	ooooooiiooooooo	ooooooiioioooooo
	ooooooiioioooooo	iiiiiioioiiiooo
	iioiiiooioiiiooo	iooioooooioooooo
	iooioooooioooooo	iooioooooooooooo
	iooioooooooooooo	ooioooooooooiooo
	oiooiiooiooiiooo	oiioiiooiooiiooo
	ooooooiooooiooooo	oiooooooooiooooo
	oiooooooiooooooo	oiooooiooooiooooo
	oiooooooiooooiooo	oooooiooooooooo
	oiooiooooiooooiooo	oiooiiooiooiooooo

For masking a plain text file or files beginning with GroupedTaxa and ending with .txt (e.g., GroupedTaxaFamily.txt) has to be provided. The format of the plain text file is simple as each new line provides the names of a clade, which shall be masked, separated by a white space. However the names have to be identical to the ones in the trees and should contain all representatives of that clade across all datasets, so all representatives might be present in all datasets. The general setup would be as follows:

TaxonA TaxonB TaxonC
TaxonE TaxonG
TaxonH TaxonD

For example,

```
Hirudo_medicinalis Helobdella_robusta Haementeria_depressa Perionyx_excavatus …
Sipunculus_nudus Themiste_lageniformes
Myzostoma_cirriferum_EST Myzostoma_seymourcollegiorum
Alvinella_pompejana Lanice_conchilega Pectinaria
Capitella_teleta Urechis_caupo
Terebratalia_transversa Bugula_neritina_EST
…
```

IMPORTANT: All input files have to be in the same folder!!!
Except for database files, which have to be placed in the folder blast/bin/Databases within the TreSpEx folder.

Options (in alphabetical order)

-aln: *name-of-file* = name of an alignment in the relaxed sequential phylip format for the combination of different partitions in -fun h

-blt: *≥ 0* = this option provides the maximal length a short branch leading to a terminal taxon can have in option –possb (default = 0), this option has to be turned on for option -possb

-bpf: *name-of-file* = name of a plain text file providing the names of files, each of which contains a list of bipartitions, which shall be independently tested by the WSRT test in –fun j; a new line provides each name:

	For example,
Testset-1	Nodes_28Sonly.txt
Testset -2	Nodes_All4Genes.txt
…	…

-db1, –db2: *name-of-database* = name of the database, which shall be used as the first or second database against which the blast searches shall be conducted; these databases should be non-redundant, that is, each orthologous sequence shall be represented by only one sequence (as for example is the case for the genome databases used by HaMStR); there are two possibilities with these options; first, a database is used, which is provided by TreSpEx already (see below) or, second, the user provides a database. In the latter case the file of the database has to be a fasta file and be placed into the folder blast/bin/Databases within the TreSpEx folder. TreSpEx will automatically recognize, if it is a protein and nucleotide database, and generate a database suitable for blast searches.

The databases of the following species are provided and the name used for –db1 or –db2 has to be exactly written as here or as in the corresponding folder in the folder blast/bin/Databases within the TreSpEx folder:

Anolis_carolinensis	*Apis_mellifera*	*Bombyx_mori*	*Bos_taurus*
Branchiostoma_floridae	*Caenorhabditis_elegans*	*Capitella_teleta*	*Ciona_intestinalis*
Danio_rerio	*Daphnia_pulex*	*Drosophila_melanogaster*	
Gallus_gallus	*Helobdella_robusta*	*Homo_sapiens*	*Lottia_gigantia*
Monodelphis_domestica	*Mus_musculus*	*Ornithorhynchus_anatinus*	
Pediculus_humanus	*Schmidtea_mediterranea*	*Taeniopygia guttata*	
Tribolium_castaneum	*Xenopus_tropicalis*		

-ediff: *≥ 0* = the difference in magnitude between the best e-value and the highest e-value, which is still considered to be part of the confidence set in –fun c, (default 5); this number has to be an integer; even if provided, there are some exceptions: if e-value of the best hit = 0.0 than only other hits, which also have an e-value of 0.0 will be considered for the confidence set or if e-value of the best hit > 0 or the magnitude of the e-value of the best hit smaller than provide magnitude (e.g. e-5) than all hits will be part of the confidence set

-evalue: *any value* = the e-value threshold for the BLAST search can be chosen as well in –fun c (default 10, optional); the input format can either 0.00001 or 1e-5 as required by the BLAST program

-fun: *a* = *a priori* screening of bootstrap support of individual gene trees as well as masking and sorting of results in accordance with certain criteria

b = *a posteriori* screening of bootstrap support of individual gene trees as well as masking and sorting of results in accordance with certain criteria

c = blast searches of alignments suspected to be affected by paralogous sequences against two reference databases

d = automatic pruning of suspicious sequences from an alignment

e = calculation of LB scores, tip-to-root distance and evolutionary rate

f = calculation of average bootstrap support of a tree

g = calculation of saturation indices based on pair-wise uncorrected p and patristic tree distances

h = generation of all possible combinations of partitions and the corresponding supermatrices and partition files as well as optionally permutated datasets of these for statistical analyses

i = summarizing bootstrap support across different datasets into a single bipartition table

j = calculation of PABA results for conflict assessment of a node-by-node and partition-by-partition basis including Wilcoxon-Signed-Rank tests and permutation significance tests

-gts: *Y, Yes, N or No* = when a masking approach for strongly supported clades in –fun a or b shall be used, the filtering has to be turned on using this option with Y or Yes (Default=No). The clades to masked have to be provided via txt-files starting with GroupedTaxa (e.g., GroupedTaxaPhylum.txt).

-ipa: = *name-of-file* = name of a plain text file providing the names of alignment files, which shall be investigated for -fun c, d and g. Be aware that for these names have to be in the same order as the list of tree files provided via -ipt, so that the name of an alignment file at that position corresponds to the name of the tree file, that was reconstructed from this alignment, at the same position in the other file provided by -ipt. The format of this file is simple and each name of an alignment file is provided in a new line:

	For example,
Alignment-name-1	21904Ali
Alignment -name-2	21912Ali
Alignment -name-3	21942Ali
Alignment -name-4	21952Ali
…	…

-ipt: = *name-of-file* = name of a plain text file providing the names of tree files, which shall be investigated for -fun a, b, c, d, e, f, g and i. Be aware that for all option also requiring the option –ipa these names have to be in the same order in both lists, so that the name of a tree file at that position corresponds to the name of the alignment file, from which the tree is was reconstructed, at the same position in the other file provided by -ipa. The format of this file is simple and each name of a tree file is provided in a new line:

	For example,
Tree-name-1	RAxML_bipartitions.21904Ali.phy_fastBoot_Out.txt
Tree-name-2	RAxML_bipartitions.21912Ali.phy_fastBoot_Out.txt
Tree-name-3	RAxML_bipartitions.21942Ali.phy_fastBoot_Out.txt
Tree-name-4	RAxML_bipartitions.21952Ali.phy_fastBoot_Out.txt
…	…

-h: *Usage* = returns the help section for usage

Input = returns the help section for input file formats

Options = returns the help section for options

Paralogy = returns the help section for the group of functions subsumed under Detection and pruning of paralogous sequences

Conflict = returns the help section for the group of functions subsumed under Detection of conflict

Mis = returns the help section for the group of functions subsumed under Miscellaneous

a, b, c, d, e, f, g, h, i or *j* = returns the help section for the function a, b, c, d, e, f, g, h, i or j, respectively

-lbnp: *≥ 0* = lower bound of numbers of partitions to be combined in -fun h, so that at least this number of partitions will be combined into a super matrix (default 0); this number has to be an integer

-lowbl: *≥ 0* = lowest factor by which the branch leading to the affected clade in option –poslb has to be at least larger than the average of all internal branch length (default = 0), this option has to be turned on for option -poslb

-lowbs: *0 – 100* = lowest BS value, which will still be considered in the screening procedure of –fun a and b (default = 95) or the lowest BS value still to be summarized in –fun f (default = 1), in both cases values between 0 and 100 are possible as long as they are integers

-ltp: *0 – 1* = the lower threshold below which a suspected sequence will be regarded as being paralogous to the other non-suspected sequences in the alignment; the value is a proportion and has to be between 0 and 1 and higher than the value provided by -upt (default 0.0); in specific, this proportion reflects the proportion of all sequences, which are not suspected of being paralogous sequences, that return the same best hit as the suspected sequence

-maxtaxa: *> 0* = maximum number of taxa, which are allowed in a clade for option -possb (default = 2), this option has to be turned on for option –possb

-maxt: *any value or N, None* = maximum threshold of support values possible or which shall be considered for the PABA calculations in -fun j; for example, the maximum threshold of bootstrap support is 100 (default = 100); the input format can either be a number or N/None if not applicable (e.g. Bremer support based values)

-mint: *any value or N, None* = minimum threshold of support values possible or which shall be considered for the PABA calculations in -fun j and the summary of bootstrap values across different dataset in -fun i; for example, the minimum threshold of bootstrap support is 0 (default = 0); the input format can either be a number or N/None if not applicable (e.g. Bremer support based values)

-**nper**: *≥ 0* = the number of permutated datasets, which shall be generated in -fun h (default 100, optional); this number has to be an integer

-**ori**: *name-of-file* = if the significance test based on permutations shall be used to test the significance of the PABA results in –fun j the name of the tabulated summary file of support values across the different datasets, in which the support values obtained from the original, non-permutated datasets were stored, has to be provided by this option; this name has also to be part of the list of summary files provided by option -tabf

-partf: *name-of-file* = name of a plain text file, which defines the partitions for -fun h and j; the format of this file is similar to the format used by RAxML, though it is not required to provide a substitution model, but it can be done. Each new line provides a new definition, which has the following general format:

name-of-substitution-model, *name-of-partition* = *definition-of-partition*

The definition of the partition can have different possibilities. The simplest definition is that the partition starts at a position and ends at one (e.g., 1-500). However, it is also possible that the partition is located in two or more parts of the datasets (e.g., 250-350, 400-500, 700-800) or that codon positions are defined (e.g., 1-333\3 for first codon position, 2-333\3 for second and 3-333\3 for third). Finally, all these can be combined into a single file, so that a file can either look like this:

```
DNA, Partition1 = 1-600
WAG, Partition2 = 601-700
DNA, Partition3 = 701-900
...
```

or like this:

```
DNA, Partition1 = 1-300\3, 2-300\3
DNA, Partition2 = 3-300\3, 701-900
DNA, Partition3 = 301-400, 461-600
DNA, Partition4 = 401-460\3
DNA, Partition5 = 402-460\3
DNA, Partition6 = 403-460\3
WAG, Partition7 = 601-700
...
```

-path: *path-to-files* = if files are not in the same folder as TreSpEx, provide the path from the root to folder with the files root by -path path-from-the-root; this option can used for all -fun

-**per**: *Y, Yes, N or No* = if permutated datasets shall be generated in –fun h or the significance of PABA results be tested using permutated datasets in –fun j, this option has to be turned on using Y or Yes (Default=No)

-poslb: *0, 1, 2, 3* = in the screening procedure the results can be sorted based on different criteria, poslb provides the position for the long branch leading to the affected clade criterion in this sorting; 0 = turned off, 1-3 = turned on and position in the sorting order is 1st, 2nd, or 3rd, respectively (default = 0); this option cannot have the same value as –possb and –possc except for 0

-possb: *0, 1, 2, 3* = in the screening procedure the results can be sorted based on different criteria, possb provides the position for the short branch leading to terminal taxon criterion in this sorting; 0 = turned off, 1-3 = turned on and position in the sorting order is 1st, 2nd, or 3rd, respectively (default = 0); this option cannot have the same value as –possc and –poslb except for 0

-possc: *0, 1, 2, 3* = in the screening procedure the results can be sorted based on different criteria, possc provides the position for the strong conflict criterion in this sorting; 0 = turned off, 1-3 = turned on and position in the sorting order is 1st, 2nd, or 3rd, respectively (default = 0); this option cannot have the same value as –possb and –poslb except for 0. If this option is used, this option –gts has also to be turned on as the masking and strong conflict are logically connected.

-ppf: *name-of-file* = name of a plain text file containing the suspected paralogous sequences, the corresponding tree, the bootstrap support value for that clade and the length of the branch leading to the clade; this option is needed for -fun c and d. The format of this file is the same as the output file generated in –fun a or b:

name-of-treefile bootstrap-support branch-length name-of-sequence name-of-sequence name-of-sequence…

For example,

RAxML_bipartitions.RPL24Ali.phy_fastBoot_Out.txt 96 1.75007055 Lanice_conchilega Eisenia_fetida
RAxML_bipartitions.22680Ali.phy_fastBoot_Out.txt 99 0.25451239 Pomatoceros_lamarckii Alvinella_pompejana
RAxML_bipartitions.22433Ali.phy_fastBoot_Out.txt 100 4.17620957 Scoloplos_armiger Sthenelais_boa Eurythoe_complanata_EST
RAxML_bipartitions.22606Ali.phy_fastBoot_Out.txt 100 0.99360951 Cerebratulus_lacteus Owenia_TS

-tabf: *name-of-file* = name of a plain text file providing the names of the tabulated summary files of the support values across the different datasets, for example as generated by -fun i, for each of which PABA calculations in -fun j shall conducted independently; usually the file with contain only a single name (the original dataset), but if the significance of the PABA results shall be assessed using permutations this file also has to contain the summary file of each single permutation; a new line provides each name:

	For example,
Summary-file-1	Original_Compiled_Bipartition_Tables.txt
Summary-file -2	Permutations_1_Compiled_Bipartition_Tables.txt
Summary-file-3	Permutations_2_Compiled_Bipartition_Tables.txt
Summary-file -4	Permutations_3_Compiled_Bipartition_Tables.txt
…	…

-tf: *name-of-file* = name of a plain text file providing the names of the taxa present in the different trees, which will be used for –fun e and i. The format of this file is simple and each taxon name is provided in a new line:

	For example,
Taxon-name-1	Eurythoe_complanata
Taxon-name-2	Terebratalia_transversa
Taxon-name-3	Arenicola_marina
Taxon-name-4	Bugula_neritina
…	…

-trf: *name-of-file* = name of a treefile containing a single tree, which provides the clades for the *a posteriori* screening approach in -fun b, in Newick format (e.g. RAxML_besttree... or RAxML_bipartition....)

-ubnp: *≥ 0* = upper bound of numbers of partitions to be combined in -fun h, so that maximal numbers of partitions to be combined into a super matrix will not be higher than this number even if more partitions could be combined (default 0); this number has to be an integer and also has to be equal to or larger than the value provided by –lbnp, but can be higher than the maximum number of defined partitions, which be then automatic maximum number of partitions to be combined into a supermatrix

-upbl: *≥ 0* = highest factor by which the branch leading to the affected clade in option –poslb has to be at least larger than the average of all internal branch length (default = 0), this option has to be turned on for option –poslb and be equal to or larger than the value of -lowbl

-upbs: *0 – 100* = highest BS value, which will still be considered in the screening procedure of –fun a and b (default = 100), values between 0 and 100 are possible as long as they are integers and equal to or larger than –lowbs

-utp: *0 – 1* = the upper threshold above which a suspected sequence will be regarded as being orthologous to the other non-suspected sequences in the alignment; the value is a proportion and has to be between 0 and 1 and higher than the value provided by -lpt (default 1.0); in specific, this proportion reflects the proportion of all sequences, which are not suspected of being paralogous sequences, that return the same best hit as the suspected sequence

-**wsrt**: *Y, Yes, N or No* = if the significance of the contribution of a partition to a certain set of bipartitions (e.g., certain clades or nodes of a tree or all nodes of a tree) shall be assessed based on the PABA results and a Wilcoxon-Signed-Rank test in –fun j, this option has to be turned on using Y or Yes (Default=No)

License

Help-Desk

If you have any problems, error-reports or other questions about TreSpEx feel free and write an email to TreSpEx@annelida.de, which is the official help desk email account for the software. For further free downloadable programs from our institute visit:

http://software.zfmk.de.

Citation

If you use TreSpEx please contact T. Struck until the manuscript addressing TreSpEx is published

Copyright

References

1. Struck, T.H. (2013). The Impact of Paralogy on Phylogenomic Studies – A Case Study on Annelid Relationships. PLoS ONE *8*, e62892.
2. Struck, T.H., Purschke, G., and Halanych, K.M. (2006). Phylogeny of Eunicida (Annelida) and Exploring Data Congruence using a Partition Addition Bootstrap Alteration (PABA) approach. Syst. Biol. *55*, 1-20.
3. Philippe, H., Brinkmann, H., Lavrov, D.V., Littlewood, D.T.J., Manuel, M., Wörheide, G., and Baurain, D. (2011). Resolving Difficult Phylogenetic Questions: Why More Sequences Are Not Enough. PLoS Biology *9*, 3.
4. Philippe, H., Derelle, R., Lopez, P., Pick, K., Borchiellini, C., Boury-Esnault, N., Vacelet, J., Renard, E., Houliston, E., Quéinnec, E., et al. (2009). Phylogenomics Revives Traditional Views on Deep Animal Relationships. Curr. Biol. *19*, 706-712.
5. Rodríguez-Ezpeleta, N., Brinkmann, H., Burger, G., Roger, A.J., Gray, M.W., Philippe, H., and Lang, B.F. (2007). Toward Resolving the Eukaryotic Tree: The Phylogenetic Positions of Jakobids and Cercozoans. Curr. Biol. *17*, 1420-1425.
6. Struck, T.H. (2007). Data congruence, paedomorphosis and salamanders. Front. Zool. *4*, 22.
7. Nosenko, T., Schreiber, F., Adamska, M., Adamski, M., Eitel, M., Hammel, J., Maldonado, M., Müller, W.E.G., Nickel, M., Schierwater, B., et al. (2013). Deep metazoan phylogeny: When different genes tell different stories. Mol. Phylogenet. Evol. *67*, 223–233.
8. Brinkman, H., and Philippe, H. (2008). Animal phylogeny and large-scale sequencing: progress and pitfalls. J. Syst. Evol. *46*, 274–286.
9. Bergsten, J. (2005). A review of long-branch attraction. Cladistics *21*, 163-193.

10. Struck, T.H., Paul, C., Hill, N., Hartmann, S., Hösel, C., Kube, M., Lieb, B., Meyer, A., Tiedemann, R., Purschke, G., et al. (2011). Phylogenomic analyses unravel annelid evolution. Nature *471*, 95–98.
11. Ebersberger, I., Strauss, S., and von Haeseler, A. (2009). HaMStR: Profile hidden markov model based search for orthologs in ESTs. BMC Evol. Biol. *9*, 157.
12. Li, L., Stoeckert, C.J., Jr., and Roos, D.S. (2003). OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res. *13*, 2178-2189.
13. Kim, K., Kim, W., and Kim, S. (2011). ReMark: an automatic program for clustering orthologs flexibly combining a Recursive and a Markov clustering algorithms. Bioinformatics *27* 1731-1733.
14. Shi, G., Peng, M.-C., and Jiang, T. (2011). MultiMSOAR 2.0: An Accurate Tool to Identify Ortholog Groups among Multiple Genomes. PLoS ONE *6*, e20892.
15. Stamatakis, A. (2006). RAxML-VI-HPC: Maximum Likelihood-based Phylogenetic Analyses with Thousands of Taxa and Mixed Models. Bioinformatics *22*, 2688–2690.
16. Lartillot, N., and Philippe, H. (2004). A Bayesian Mixture Model for Across-Site Heterogeneities in the Amino-Acid Replacement Process. Mol. Biol. Evol. *21*, 1095-1109.
17. Thornton, J.W., and DeSalle, R. (2000). A New Method to Localize and Test the Significance of Incongruence: Detecting Domain Shuffling in the Nuclear Receptor Superfamily. Syst. Biol. *49*, 183-201.
18. Farris, J.S., Kallersjo, M., Kluge, A.G., and Bult, C. (1995). Constructing a significance test for incongruence. Syst. Biol. *44*, 570–572.
19. Templeton, A.R. (1983). Phylogenetic inference from restriction site endonuclease cleavage site maps with particular reference to the human and apes. Evolution *37*, 221-244.
20. Macey, J.R., Schulte, J.A., Larson, A., Tuniyev, B.S., Orlov, N., and Papenfuss, T.J. (1999). Molecular Phylogenetics, tRNA evolution, and historical biogeography in anguid lizards and related taxonomic families. Mol. Phylogenet. Evol. *12*, 250-272.
21. Whitlock, B.A., and Baum, D.A. (1999). Phylogeny of cacao (*Theobroma cacao* L., Sterculiaceae) and its wild relatives based on sequences of the nuclear-encoded gene *VICILIN*. Syst. Bot. *24*, 128-138.
22. Lee, M.S.Y. (2000). Tree Robustness and Clade Significance. Syst. Biol. *49*, 829-836.
23. Lee, M.S.Y., and Hugall, A.F. (2003). Partitioned Likelihood Support and the Evaluation of Data Set Conflict. Syst. Biol. *52*, 15-22.
24. Struck, T.H. (2011). Direction of evolution within Annelida and the definition of Pleistoannelida. J. Zool. Syst. Evol. Res. *49*, 340-345.