# SuperTRI3

## A) Short description of the method
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

The SuperTRI method is based on the branch support analyses of the independent data sets, in which the reliability of the nodes is assessed using three measures: the supertree Bootstrap percentage (SBP) and two other values calculated from the separate analyses: the mean branch support (mean Bootstrap percentage [MBP] or mean posterior probability [MPP]) and the reproducibility index (Ropiquet et al., 2009). The SuperTRI approach shows less sensitivity to the phylogenetic methods and models, and it is more accurate to interpret the relationships among taxa.

**supertri3.py**
Copyright (C) 2009-2024 Anne Ropiquet, Blaise Li, Thanina Chabane and Alexandre Hassanin

supertri3.py is a python script that will help you producing a matrix representation weighted by bootstrap proportions or bayesian posterior probabilities. Such a matrix representation can be used to produce supertrees.

supertri3.py also computes the support indices described in Ropiquet et al. (2009): reproducibility and mean robustness (or mean bayesian posterior probability) and may report these indices on nexus trees.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA

supertri3.py is a python script. To be able to run it, you must first check wether python is installed.

In MacOSX there is a command-line
terminal called "Terminal"; it should be in the "Utilities" directory of the "Applications" directory.

## B) Choose a place where to install supertri.py
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

It is advisable to put files in a place where you you can find it later. There are usually some conventions on where to put executable files. For example, on an UNIX-like operating system (UNIX, Linux, MacOSX), executables are often placed in directories ending in "bin" (bin for "binary": that is, programs compiled in machine language, note that supertri3.py is not in machine language; you may open this python script in a text editor and see human-readable commands, python is the program that will translate these commands into machine language).

There are system-wide bin directories, like /usr/local/bin, that are usually in the PATH of every user (this means that programs in such a place will be found and executed when the user types the name of the program in a command-line terminal). To place a program in /usr/local/bin you need to have administrative rights. Under MacOSX, it is possible that directories like /usr/local/bin are not visible from the finder, but only from the command-line terminal. In such a case, you will have to use the command "cp" to copy supertri3.py to the place you want.

If you are not allowed to place a program in /usr/local/bin, you may place the program in a personal bin directory. A good place would be a bin directory in your home directory. For example, if your username is dupont and you are in a MacOSX system, this would be /Users/dupont/bin. On a Linux system, this would be /home/dupont/bin.

## C) There are several ways to execute a python script
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Once you have placed supertri3.py in a suitable place, if you want to be able to call the command directly, you'll have to check wether that place is in your PATH or not (see A) 2)). You may also call python and tell it to run the supertri3.py script.

1) The directory containing supertri3.py is in your PATH

This is probably the most convenient situation.

Open a command-line terminal.

Go to the directory where you placed supertri.py, using the command "cd" ("change directory"):

```
----- MacOSX-like example -----
Welcome to Darwin !
Computer:~ dupont$ cd /Users/dupont/bin
Computer:~/bin dupont$
-----
```

Make the script executable, using the command "chmod":

```
-----
Computer:~/bin dupont$ chmod +x supertri3.py
Computer:~/bin dupont$
-----
```

Now you can call supertri3.py in command-line from any directory (provided your PATH contains the directory where supertri3.py has been placed):

```
-----
Computer:~/bin dupont$ cd
Computer:~ dupont$ supertri3.py

usage: supertri3.py [options] <arguments>
available options:
[list of options here]
In interactive mode, the user will be prompted for the arguments,
otherwise, the arguments should be given in the command line.

Traceback (most recent call last):
  File "/Users/dupont/bin/supertri.py", line 995, in ?
    main()
  File "/Users/dupont/bin/supertri.py", line 788, in main
    raise Exception("No dataset names provided. Either use option -i or provide
a file with option --marklist.")
Exception: No dataset names provided. Either use option -i or provide a file
with option --marklist.
-----
```

Well, it seems to work. Go to D).

2) You want to call python explicitly

Open a command-line terminal.

Go to the directory where you placed supertri.py, using the command "cd":

```
----- MacOSX-like example -----
Welcome to Darwin !
Computer:~ dupont$ cd /Users/dupont/bin
Computer:~/bin dupont$
-----
```

Tell python to execute the script:

```
-----
Computer:~/bin dupont$ python supertri3.py

usage: supertri3.py [options] <arguments>
available options:
[list of options here]
In interactive mode, the user will be prompted for the arguments,
otherwise, the arguments should be given in the command line.

Traceback (most recent call last):
  File "supertri3.py", line 995, in ?
    main()
  File "supertri3.py", line 788, in main
    raise Exception("No dataset names provided. Either use option -i or provide
a file with option --marklist.")
Exception: No dataset names provided. Either use option -i or provide a file
with option --marklist.
-----
```

3) You want to execute the script from anywhere, but it is not in your PATH

Open a command-line terminal.

Go to the directory where you placed supertri3.py, using the command "cd":

```
----- MacOSX-like example -----
Welcome to Darwin !
Computer:~ dupont$ cd /Users/dupont/bin
Computer:~/bin dupont$
-----
```

Make the script executable, using the command "chmod":

```
-----
Computer:~/bin dupont$ chmod +x supertri3.py
Computer:~/bin dupont$
-----
```

Go to the directory where you placed your analyses results, using the command "cd":

```
-----
Computer:~/bin dupont$ cd /Users/dupont/Documents/Analyses/Smurfs/bootstrap
Computer:~/Documents/Analyses/Smurfs/bootstrap dupont$
-----
```

Execute the script, calling it with its absolute path:

```
-----
Computer:~/Documents/Analyses/Smurfs/bootstrap dupont$
/Users/dupont/bin/supertri3.py

usage: supertri3.py [options] <arguments>
available options:
[list of options here]
In interactive mode, the user will be prompted for the arguments,
otherwise, the arguments should be given in the command line.

Traceback (most recent call last):
  File "/Users/dupont/bin/supertri.py", line 995, in ?
    main()
  File "/Users/dupont/bin/supertri.py", line 788, in main
    raise Exception("No dataset names provided. Either use option -i or provide
a file with option --marklist.")
Exception: No dataset names provided. Either use option -i or provide a file
with option --marklist.
-----
```

## D) Input files
^^^^^^^^^^^^^^

A preliminary piece of advice:

When working with command-line tools, it is usually better not to have files with names containing spaces, accentuated letters or special characters other than underscore ("_"). Spaces are used to separate commands and arguments. Special characters may have special meanings. Accentuated letters are not always handled correctly.

To produce a valid nexus matrix representation of your results, weighted by support values, supertri3.py needs at least the following:

- files describing the bipartitions with "." and "*" and the support values, produced by the analyses of your different datasets.
- a file containing the taxon names, in the same order as in the matrix used for the analyses, without blanks in the names, and with a name on each line of the file.
- if there are taxa missing from one or more datasets, prepare for each dataset a file containing the names of the taxa missing for this dataset.

The results files should be named the following way:
Dataset.parts for the results of a MrBayes analysis, where Dataset is the name of the dataset that was analysed.
Dataset.log for the log of a PAUP bootstrap analysis, where Dataset is the name of the dataset that was analysed.

Since the description of bipartitions with "." and "*" is made without taxon names, the analyses should all have been done whith the taxa in the same order in the matrix. Otherwise, the representations of the bipartitions from two different analyses will not be comparable. This order is given to supertri.py by the file with the names of the taxa.

The files for the missing taxa should contain the taxa names as they appear in the list of the taxa. These files should be named the following way:
Dataset.abs, where Dataset is the name of the dataset lacking the taxa in the file.

It is recommended, but optional to prepare a file with the names of the datasets, without blanks in the names of the datasets, and with one name per line. These names should correspond to the prefixes of the files containing the results.

It is advisable not to have other files than those concerned with the analyses in the directory where you put the data for supertri3.py.

## E) Executing supertri3.py
^^^^^^^^^^^^^^^^^^^^^^^^

As many command-line tools, supertri3.py works with a system of options.

Suppose the names of the taxa are written in a file named "taxons". To indicate supertri3.py to use this file, the command will be called with the --taxlist option:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons
-----
```

This is not enough to make it work. If you want to provide a list of datasets on the command line, with a file named "genes", you will have to use the --datasets option:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons --datasets genes
-----
```

You may also run the program in interactive mode, with the -i option, so that you can provide the names of the datasets interactively:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons -i
-----
```

You may also specify the type of results files with the --suffix option:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons --datasets genes --
suffix .log
-----
```

You can chose the name of the matrix to write with the option -o:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons --datasets genes --
suffix .log -o MRP.nex

-----
```

The default name will be composed by the names of the datasets followed by ".mrpp.nex" for MrBayes data or ".mrbp.nex" for bootstrap log data.
You can chose the root of the output trees with the option --root:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons --datasets genes --
suffix .log -o MRP.nex --root The_outgroup_taxon
-----
```

You must ensure that the name you give is present in the file listing the taxa, but not in any .abs file.

You can give trees on which to map the indices with the option --intree:

```
-----
Computer:~ dupont$ python3 supertri3.py --taxlist taxons --datasets genes --
suffix .log -o MRP.nex --root The_outgroup_taxon --intree synthesistree.tre



-----
```

The trees you give with this option must be in nexus format and contain the names of the taxa in the parenthesized representation of the tree; not in a translation block. You can provide several trees in the same file, or call the option several times with different file names.

If you want to map the indices on a tree obtained from the matrix representation produced by a first run of supertri.py, you'll have to run supertri3.py again, with the --intree option, with the previously used parameters.

The indices will be written as node labels in two separate nexus trees. They will also be written in a .tgf file for graphical export of .svg or .eps trees. In the .tgf file, the first index above the branch is the mean support and the second is the reproducibility index.

## F) Miscellaneous remarks
^^^^^^^^^^^^^^^^^^^^^^^^

This program was not written by a professional programmer. We hope it works and we hope the code is clear enough to allow you to modify it if you know python.

If python works and you are able to launch supertri3.py but you still have problems running this program, check carefully that your files conform to the requirements of part D).

There may also be problems due to file end-of-line coding.

Before reporting bugs, note the error messages, if any, and try to also determine the version of python and the version of supertri3.py that you are using. Please also indicate the type of operating system you are using (Windows, Mac, Linux, with details about the particular version of the operating system).

## G) References
^^^^^^^^^^^^^

Citation

Ropiquet A., Li B. & Hassanin A. (2009) SuperTRI: a new approach based on branch support analyses of multiple independent data sets for assessing reliability of phylogenetic inferences. Comptes Rendus Biologies 332: 832–847.

References using SuperTRI

Hassanin A. & Rambaud O. (2023) Retracing phylogenetic, host and geographic origins of coronaviruses with coloured genomic bootstrap barcodes: SARS-CoV and SARS-CoV-2 as case studies. Viruses 15: 406. doi: 10.3390/v15020406

Sabroux R., Corbari L. & Hassanin A. (2023) Phylogeny of sea spiders (Arthropoda: Pycnogonida) inferred from mitochondrial genome and 18S ribosomal RNA gene sequences. Molecular Phylogenetics and Evolution 182: 107726. doi:10.1016/j.ympev.2023.107726

Hassanin A., Rambaud O. & Klein D. (2022) Genomic bootstrap barcodes and their application to study the evolution of sarbecoviruses. Viruses 14: 440. doi: 10.3390/v14020440

Curaudeau M., Rozzi R. & Hassanin A. (2021) The genome of the lowland anoa (Bubalus depressicornis) illuminates the origin of river and swamp buffalo. Mol. Phylogenetics Evol. 161; doi: 10.1016/j.ympev.2021.107170

Hassanin A., Veron G., Ropiquet A., Jansen van Vuuren B., Lécu A., Goodman S.M., Haider J. & Nguyen T.T. (2021). Evolutionary history of Carnivora (Mammalia, Laurasiatheria) inferred from mitochondrial genomes. PloS one 16(2): e0240770. doi: 10.1371/journal.pone.0240770

Hassanin A., Bonillo C., Tshikung D., Pongombo Shongo C., Pourrut X., Kadjo B., Nakouné E., Tu V.T., Prié V. & Goodman S.M. (2020) Phylogeny of African fruit bats (Chiroptera, Pteropodidae) based on complete mitochondrial genomes. J Zool Syst Evol Res. 58: 1395–1410. doi: 10.1111/jzs.12373

Petzold A. & Hassanin A. (2020) A comparative approach for species delimitation based on multiple methods of multi-locus DNA sequence analysis: A case study of the genus Giraffa (Mammalia, Cetartiodactyla). PLoS ONE 15 (2): e0217956. doi: 10.1371/journal.pone.0217956

Tu V.T., Hassanin A., Furey N.M., Son N.T. & Csorba G. (2018) Four species in one: multigene analyses reveal phylogenetic patterns within Hardwicke’s woolly bat, Kerivoula hardwickii-complex (Chiroptera, Vespertilionidae) in Asia. Hystrix It. J. Mamm. 29:111-121.

Hassanin A., Houck M.L., Tshikung D., Kadjo B., Davis H. & Ropiquet A. (2018) Multi-locus phylogeny of the tribe Tragelaphini (Mammalia, Bovidae) and species delimitation in bushbuck: Evidence for chromosomal speciation mediated by interspecific hybridization. Mol. Phylogenetics Evol. 129: 96-105.

Hassanin A., Colombo R., Gembu G.-C., Merle M., Tu V.T., Görföl T., Akawa P.M., Csorba G., Kearney T., Monadjem A. & Ing R.K. (2018) Multilocus phylogeny and species delimitation within the genus Glauconycteris (Chiroptera, Vespertilionidae), with the description of a new bat species from the Tshopo Province of the Democratic Republic of the Congo. Journal of Zoological Systematics and Evolutionary Research 56: 1–22.

Tu V.T., Csorba G., Ruedi M., Furey N.M., Son N.T., Thong V.D., Bonillo C. & Hassanin A. (2017) Comparative phylogeography of bamboo bats of the genus Tylonycteris (Chiroptera, Vespertilionidae) in Southeast Asia. European Journal of Taxonomy 274: 1-38.

Hassanin A., Nesi N., Marin J., Kadjo B., Pourrut X., Leroy E., Gembu G.C., Akawa P.M., Ngoagouni C., Nakouné E., Ruedi M., Tshikung D., Pongombo Shongo C. & Bonillo C. (2016). Comparative phylogeography of African fruit bats (Chiroptera, Pteropodidae) provides new insights into the outbreak of Ebola virus disease in West Africa, 2014–2016. Comptes Rendus Biologies 339: 517-528.

Hassanin A., Khouider S., Gembu G.C., Goodman S.M., Kadjo B., Nesi N., Pourrut X., Nakouné E. & Bonillo C. (2015) The comparative phylogeography of fruit bats of the tribe Scotonycterini (Chiroptera, Pteropodidae) reveals cryptic species diversity related to African Pleistocene forest refugia. Comptes Rendus Biologies 338: 197-211.

Hassanin A., An J., Ropiquet A., Nguyen T.T. & Couloux A. (2013) Combining multiple autosomal introns for studying shallow phylogeny and taxonomy of Laurasiatherian mammals: Application to the tribe Bovini (Cetartiodactyla, Bovidae). Mol. Phylogenetics Evol. 66: 766–775.

Nesi N., Kadjo B., Pourrut X., Leroy E., Pongombo Shongo C., Cruaud C. & Hassanin A. (2013). Molecular systematics and phylogeography of the tribe Myonycterini (Mammalia, Pteropodidae) inferred from mitochondrial and nuclear markers. Mol. Phylogenetics Evol. 66: 126–137.

Hassanin A., Delsuc F., Ropiquet A., Hammer C., Jansen van Vuuren B., Matthee C., Ruiz-Garcia M., Catzeflis F., Areskoug V., Nguyen T.T. & Couloux A. (2012) Pattern and timing of diversification of Cetartiodactyla (Mammalia, Laurasiatheria), as revealed by a comprehensive analysis of mitochondrial genomes. Comptes Rendus Biologies 335: 32-50.

Hassanin A., Bonillo C., Nguyen B.X. & Cruaud C. (2010) Comparisons between mitochondrial genomes of domestic goat (Capra hircus) reveal the presence of numts and multiple sequencing errors. Mitochondrial DNA 21: 68-76.