TraitRateProp – manual and code documentation

1. **Overview** – the purpose of the code is to infer associations between transitions in the character state and the rate of sequence evolution. Inference can be performed for three models: ‘null’, ‘TR’, and ‘TRP’.

   Before going into the technical details, the program takes as input the following:

   1.1. A rooted ultrametric phylogentic tree with branch lengths (Newick)
   1.2. A multiple sequence alignment (MSA) of the sequence data of the extant species (Fasta)
   1.3. The character states of the extant species coded as either ‘0’ or ‘1’ (Fasta)

   The program outputs result files with the inferred model parameter values and the log-likelihood of the fitted models.

   The full details of the TraitRateProp model are given in its manuscript (Levy Karin E., Wicke S., Pupko T., and Mayrose I. *An integrated model of phenotypic trait changes and site-specific sequence evolution*. 2017). The details concerning its implementation are given in the appendix. This manual and code documentation assumes familiarity with the information therein.

2. **Source code and executable** – in the Rar file we provide:
   - The TraitRateProp source code
   - A pre-compiled program for UNIX
   - Template control files

3. **Compilation** – the compiler version used by us is gcc 6.2.0. In order to compile on UNIX you will need to perform the following steps:
   1. Go to directory libs/phylogeny/ by typing: "cd DOWNLOAD_DIR/libs/phylogeny/"
   2. To compile the library type: "make doubleRep"
   3. Go to the source directory by typing: "cd ../../programs/traitRateProp/"
   4. Compile the program by typing: "make doubleRep".

   The executable traitRate.doubleRep will be created under: programs/ traitRateProp /

4. **Running TraitRateProp** – the program is run with a configuration file:
   traitRate.doubleRep CONFIGURATION_FILE

5. **Running a toy example** – As a first step we recommend running a toy example, following these steps:

   5.1. Create a directory for the example (mkdir EXAMPLE_DIRECTORY)
   5.2. Save the example input MSA file to EXAMPLE_DIRECTORY
   5.3. Save the example input trait states file to EXAMPLE_DIRECTORY
   5.4. Save the example input tree file to EXAMPLE_DIRECTORY
   5.5. Save the example TraitRateProp configuration file to EXAMPLE_DIRECTORY
   5.6. Edit the example TraitRateProp configuration file by replacing "FULL_PATH_TO_YOUR_EXAMPLE_DIRECTORY" with the full path to the example directory you created
   This edit will require 5 changes in the example TraitRateProp configuration file
   5.7. Run SpartaABC with the edited configuration file:
   5.8. traitRate.doubleRep EXAMPLE_DIRECTORY/ TRP_configuration_file.txt

   The outputs of the run will be written in FULL_PATH_TO_YOUR_EXAMPLE_DIRECTORY/result/

6. **Input configuration –** the code is run with a configuration file in which input parameters are set. The parameters of interest for the user are described in the table below. In addition, we provide template configuration files (see next section of this document).

Parameter group	**Parameter name**	**Description**	**Possible values**	**Default value**
Program run mode	_mainType	TraitRateProp run mode. Either both “null” and “alternative” models are	“Optimize_Model” (null & alternative)	

		computed or just the “alternative” model is computed. In case both are computed, the null results serve as one starting point for the alternative model	“Optimize_Model_Alter_Only” alternative) Other options exist for debug and simulation purposes (not for users)	
Input files	_treeFile	An ultrametric newick format tree	a full path to the file	
	_characterFile	Character states for all extant species in fasta format	a full path to the file	
	_seqFile	MSA of all extant species in fasta format	a full path to the file	
Output files and directories	_outDir	Directory where the output should be written	a full path to the directory	
	_outFileNullParams	The null model output	a full path to the file	
	_outFile	The alternative model output	a name (will be created under the output directory)	
	_logFile	The log output	a name (will be created under the output directory)	log.txt
	_LLPerPositionFile	The likelihood of each position as trait-dependent and as trait-independent	a name (will be created under the output directory)	
	_scaledTreeFile	The final sum of branch lengths dictates a factor by which the ultrametric tree is scaled. The final character model parameters dictate a collection of stochastic mappings. Each of these mappings is used with the final *r* parameter to stretch the final scaled ultrametric tree (no longer ultrametric after the stretch). These stretched versions of the scaled ultrametric tree are averaged and the result is written to the file defined by _scaledTreeFile	a name (will be created under the output directory)	scaled.tree
What to optimize	_bGridStartPoints	Should (*p*,*r*) combinations for starting points be sampled from a grid or randomly. If either *p* or *r* are not optimized – set to 0	0 - randomly,1 – by grid	1
	_bOptCharModel	Should character model parameters be optimized	0 – don’t optimize,1 - optimize	1
	_bOptProportion	Should *p* (proportion) parameter be optimized	0 – don’t optimize,1 - optimize	1
	_bOptRelRate	Should *r* (relative rate) parameter be optimized	0 – don’t optimize,1 - optimize	1
	_bOptSeqModel	Should sequence model parameters be optimized	0 – don’t optimize,1 - optimize	1
	_bScaleTree	Should sum of branch lengths be optimized (search for a factor by which all branches are multiplied)	0 – don’t optimize,1 - optimize	1
How to optimize	_optimizeIterNum	Number of optimization iterations in each round	Non-negative integers separated by commas	0,2,5
	_optimizePointsNum	Number of points to optimize in each round	Non-negative integers separated by commas	10,3,1
	_optimizeStrategies	Optimization strategy in each round (1 = heuristic, 0 = exhaustive)	{0,1} separated by commas	1,1,1
	_stochasicMappingIterations	Number of stochastic mappings to be used	Positive integer	100
Setting	_charModelParam1	Character model	[0,1]	0.5

parameter values		π_1 parameter value. If given, will be used for all starting points. If character model is not optimized – will be used as value		
	_charModelParam2	Character model μ parameter value. If given, will be used for all starting points. If character model is not optimized – will be used as value	(0,MAX_DOUBLE)	1.0
	_gammaCategories	Number of discrete rate categories to use	Positive integer	4
	_gammaParam	The α parameter of the gamma distribution of rate variation among sequence sites. If given, will be used for all starting points. If sequence model is not optimized – will be used as value	(0,MAX_DOUBLE)	1.0
	_proportion	The *p* (proportion of positions associated with the trait) parameter. If given, will be used for all starting points. If proportion is not optimized – will be used as value	[0,1]	1.0
	_relRate	The *r* (relative rate) parameter. If given, will be used for all starting points. If proportion is not optimized – will be used as value	(0, MAX_DOUBLE)	1.0
	_relRateMaxVal	The maximal r value to sample from when selecting starting points. Values are selected between 1/r and r (as this is a multiplicative factor, the range is first log-transformed then sampled uniformly and then transformed back)	(0, MAX_DOUBLE)	4.0
	_seqModelParam1	The κ parameter (transition / transversion) of the HKY85 model. If given, will be used for all starting points. If sequence model is not optimized – will be used as value	(0,MAX_DOUBLE)	1.0
	_sequenceType	The kind of sequence data to analyze (DNA or AA).	{NON_CODING,PROTEIN}	NON_CODING
	_seqModelType	The kind of sequence model to use.	{HKY,JTT}	HKY
	_treeLength	Sum of branch lengths. If given, will be used for all starting points. If tree length is not optimized – will be used as value	{-1.0,(0,MAX_DOUBLE)}	-1.0 (no set value)

7. **Template configuration files –** we provide the following inference templates:
    1. Inferring with the TR model (proportion parameter is set to 1) and with the null model
    2. Inferring with the TRP model (proportion is free) and the null model
    3. Inferring with the TRP model (proportion is free)

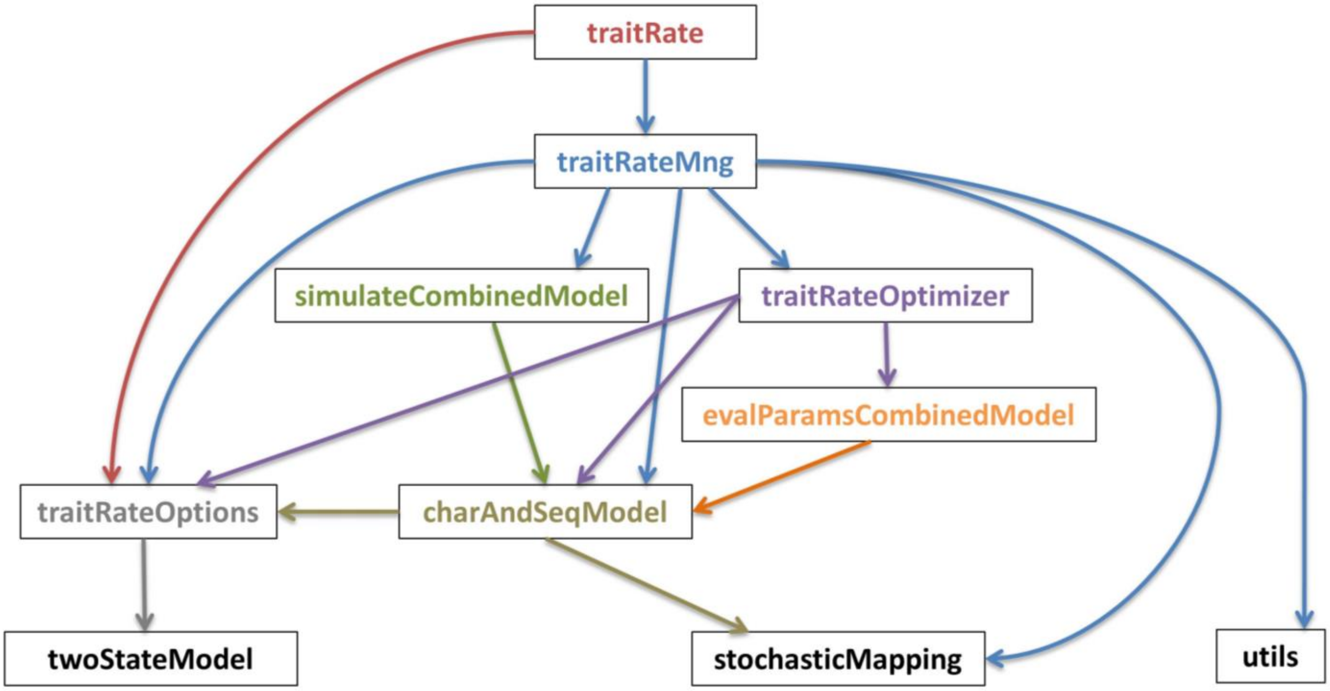
If you are not interested in comparing the TR model to the TRP model, we recommend running option number 2 (comparing the TRP model to the null model).

In addition, if you work with option 3, you could provide the resulting inferred parameters of the TR model as a starting point to the TRP model. In order to this, you'll need to obtain the parameter estimates from the result file of the run with template number 1. For example:

```
_charModelParam1 0.641014
_charModelParam2 15.4164
_relRate 1.1
_proportion 1
_treeLength 0.7932
_gammaParam 0.571747
_seqModelParam1 1.01107
```

8. **Outputs –** TraitRateProp generates three output files of interest:
   - 8.1. The alternative results file (set in **_outFile**) – contains information about the parameter estimations under the alternative model (TR or TRP) and the log-likelihood scores of the null model (model 0) and the alternative model. These scores can be used for the chi-sq LRT.
   - 8.2. The null results file (set in **_outFileNullParams**) – contains the null model parameter estimations.
   - 8.3. The position likelihoods file (set in **_LLPerPositionFile**) – in the TRP alternative model, the likelihood of each position is computed once with no association between the rate and the phenotypic trait and once with such an association. This file contains these computations for each position and the ratio between them (= the Bayes factor).

9. **Code structure scheme –** only TraitRateProp internal dependencies are presented:



10. **Code modules main purpose**:
    - 10.1. **utils**:
      <u>Purpose</u>: Provide functionality, such as checking tree and MSA have matching taxa.
    - 10.2. **stochasticMapping**:
      <u>Purpose</u>: Generate stochastic mappings and related functionality.

10.3. **twoStateModel**:
Purpose: Character (trait) model representation and functionality.

10.4. **traitRateOptions**:
Purpose: Manage input configuration file and default values

10.5. **CharAndSeqModel**:
Purpose: Represent a joint object with two stochastic processes; that of the sequence and that of the character as well as the tree and the relative rate and proportion parameters. Handle all likelihood computations and tree scaling procedures given all of its defined parameters.

10.6. **evalParamsCombinedModel**:
Purpose: Provide wrappers for the various likelihood computation modes of CharAndSeqModel. This is needed for Brent optimization scheme.

10.7. **simulateCombinedModel**:
Purpose: Handle character and sequence simulation given model parameters. It is used for internal purposes only (simulation study, debug) and nit intended for users.

10.8. **traitRateOptimizer**:
Purpose: Manage optimization scheme: “null”, “alternative” or both. In case of “alternative” optimization, handle the optimization rounds in terms of starting points, strategies and maximum allowed iterations. When optimizing each point, a loop of optimizations is performed (according to the parameters set to be optimized): the relative rate and proportion parameters, the sequence model, the tree length, and the character model. When changes to the tree length or the character model parameters occur, a new set of stochastic mappings is generated.

10.9. **traitRateMng**:
Purpose: Initialize models, set up and manage the required optimization scheme and write the required outputs.

10.10. **traitRate**:
Purpose: Handle the program run mode. If valid, call the required traitRateMng functionality.