Package ‘secsse’

February 12, 2021

Type Package

Title Several Examined and Concealed States-Dependent Speciation and Extinction

Version 2.0.7

Date 2021-02-10

License GPL-3

Description Simultaneously infers state-dependent diversification across two or more states of a single or multiple traits while accounting for the role of a possible concealed trait. See Herrera-Alsina et al. 2019 Systematic Biology 68: 317-328 <DOI:10.1093/sysbio/syy057>.

Depends R (>= 3.5.0)

Imports utils, DDD (>= 4.0), ape, foreach, doParallel, apTreeshape, phylobase, geiger, deSolve

Suggests diversitree, phytools, testthat, testit, knitr, rmarkdown

Enhances doMC

NeedsCompilation yes

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.1.1

Author Leonel Herrera Alsina [aut],
Paul van Els [aut],
Rampal S. Etienne [cre, aut] (<https://orcid.org/0000-0003-2142-7612>)

Maintainer Rampal S. Etienne <`r.s.etienne@rug.nl`>

Repository CRAN

Date/Publication 2021-02-12 16:50:09 UTC

R topics documented:

`cla_id_paramPos` *Parameter structure setting for cla_secsse*

Description

It sets the parameters (speciation, extinction and transition) ids. Needed for ML calculation with cladogenetic options (cla_secsse_ml)

Usage

```
cla_id_paramPos(traits, num_concealed_states)
```

Arguments

`traits`	vector with trait states, order of states must be the same as tree tips, for help, see vignette.
`num_concealed_states`	number of concealed states, generally equivalent to number of examined states.

Value

A list that includes the ids of the parameters for ML analysis.

Examples

```
traits <- sample(c(0,1,2), 45,replace = TRUE) #get some traits
num_concealed_states <- 3
param_posit <- cla_id_paramPos(traits,num_concealed_states)
```

cla_secsse_loglik *Likelihood for SecSSE model*

Description

Logikelihood calculation for the cla_SecSSE model given a set of parameters and data

Usage

```
cla_secsse_loglik(
  parameter,
  phy,
  traits,
  num_concealed_states,
  use_fortran = TRUE,
  methode = "ode45",
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  run_parallel = FALSE,
  setting_calculation = NULL,
  setting_parallel = NULL,
  see_ancestral_states = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
 func = ifelse(is_complete_tree, "cla_secsse_runmod_ct", ifelse(use_fortran == FALSE,
    cla_secsse_loglik_rhs, "cla_secsse_runmod"))
)
```

Arguments

`parameter`	list where the first is a table where lambdas across different modes of speciation are shown, the second mus and the third transition rates.
`phy`	phylogenetic tree of class phylo, ultrametric, fully-resolved, rooted and with branch lengths.
`traits`	vector with trait states, order of states must be the same as tree tips, for help, see vignette.
`num_concealed_states`	number of concealed states, generally equivalent to number of examined states.
`use_fortran`	Should the Fortran code for numerical integration be called? Default is TRUE.
`methode`	Solver for the set of equations, default is "ode45".
`cond`	condition on the existence of a node root: "maddison_cond","proper_cond"(default). For details, see vignette.

`root_state_weight`	the method to weigh the states:"maddison_weights","proper_weights"(default) or "equal_weights". It can also be specified the root state:the vector c(1,0,0) indicates state 1 was the root state.
`sampling_fraction`	vector that states the sampling proportion per trait state. It must have as many elements as trait states.
`run_parallel`	should the routine to run in parallel be called?
`setting_calculation`	argument used internally to speed up calculation. It should be leave blank (default : setting_calculation = NULL)
`setting_parallel`	argument used internally to set a parallel calculation. It should be left blank (default : setting_parallel = NULL)
`see_ancestral_states`	should the ancestral states be shown? Deafault FALSE
`loglik_penalty`	the size of the penalty for all parameters; default is 0 (no penalty)
`is_complete_tree`	whether or not a tree with all its extinct species is provided
`func`	function to be used in solving the ODE system. Currently only for testing purposes.

Value

The loglikelihood of the data given the parameters

Note

To run in parallel it is needed to load the following libraries when windows: apTreeshape, doparallel and foreach. When unix, it requires: apTreeshape, doparallel, foreach and doMC

Examples

```
rm(list=ls(all=TRUE))
library(secsse)
library(DDD)
library(deSolve)
#library(diversitree)
library(apTreeshape)
library(foreach)
set.seed(13)
phylotree <- ape::rcoal(12, tip.label = 1:12)
traits <- sample(c(0,1,2),ape::Ntip(phylotree),replace=TRUE)
num_concealed_states <- 3
sampling_fraction <- c(1,1,1)
methode <- "ode45"
phy <- phylotree
# the idparlist for a ETD model (dual state inheritance model of evolution)
# would be set like this:
```

```
idparlist <- cla_id_paramPos(traits,num_concealed_states)
lambd_and_modeSpe <- idparlist$lambdas
lambd_and_modeSpe[1,] <- c(1,1,1,2,2,2,3,3,3)
idparlist[[1]] <- lambd_and_modeSpe
idparlist[[2]][] <- 0
masterBlock <- matrix(4,ncol=3,nrow=3,byrow=TRUE)
diag(masterBlock) <- NA
idparlist [[3]] <- q_doubletrans(traits,masterBlock,diff.conceal = FALSE)
# Now, internally, clasecsse sorts the lambda matrices, so they look like:
prepare_full_lambdas(traits,num_concealed_states,idparlist[[1]])
# which is a list with 9 matrices, corresponding to the 9 states (0A,1A,2A,0B,etc)
# if we want to calculate a single likelihood:
parameter <- idparlist
lambd_and_modeSpe <- parameter$lambdas
lambd_and_modeSpe[1,] <- c(0.2,0.2,0.2,0.4,0.4,0.4,0.01,0.01,0.01)
parameter[[1]] <- prepare_full_lambdas(traits,num_concealed_states,lambd_and_modeSpe)
parameter[[2]] <- rep(0,9)
masterBlock <- matrix(0.07,ncol=3,nrow=3,byrow=TRUE)
diag(masterBlock) <- NA
parameter [[3]] <- q_doubletrans(traits,masterBlock,diff.conceal = FALSE)
cla_secsse_loglik(parameter, phy, traits, num_concealed_states,
                 use_fortran = FALSE, methode = "ode45", cond = "maddison_cond",
                 root_state_weight = "maddison_weights", sampling_fraction,
                 run_parallel = FALSE, setting_calculation = NULL,
                 setting_parallel = NULL, see_ancestral_states = FALSE,
                 loglik_penalty = 0)
# LL = -37.8741
```

cla_secsse_ml *Maximum likehood estimation for (SecSSE)*

Description

Maximum likehood estimation under Several examined and concealed States-dependent Speciation and Extinction (SecSSE) with cladogenetic option

Usage

```
cla_secsse_ml(
  phy,
  traits,
  num_concealed_states,
  idparslist,
  idparsopt,
  initparsopt,
  idparsfix,
  parsfix,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  tol = c(1e-04, 1e-05, 1e-07),
  maxiter = 1000 * round((1.25)^length(idparsopt)),
  use_fortran = TRUE,
  methode = "ode45",
  optimmethod = "simplex",
  num_cycles = 1,
  run_parallel = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  func = "cla_secsse_runmod",
  verbose = (optimmethod == "subplex")
)
```

Arguments

`phy`	phylogenetic tree of class phylo, ultrametric, rooted and with branch lengths.
`traits`	a vector with trait states for each tip in the phylogeny.
`num_concealed_states`	number of concealed states, generally equivalent to the number of examined states in the dataset.
`idparslist`	overview of parameters and their values.
`idparsopt`	id of parameters to be estimated.
`initparsopt`	initial guess of the parameters to be estimated.
`idparsfix`	id of the fixed parameters.
`parsfix`	value of the fixed parameters.
`cond`	condition on the existence of a node root: "maddison_cond","proper_cond"(default). For details, see vignette.
`root_state_weight`	the method to weigh the states:"maddison_weights","proper_weights"(default) or "equal_weights". It can also be specified the root state:the vector c(1,0,0) indicates state 1 was the root state.
`sampling_fraction`	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states.
`tol`	maximum tolerance. Default is "c(1e-04, 1e-05, 1e-05)".
`maxiter`	max number of iterations. Default is "1000 *round((1.25)^length(idparsopt))".
`use_fortran`	Should the Fortran code for numerical integration be called? Default is TRUE.
`methode`	method used for integration calculation. Default is "ode45".
`optimmethod`	method used for optimization. Default is "simplex".
`num_cycles`	number of cycles of the optimization (default is 1).
`run_parallel`	should the routine to run in parallel be called?
`loglik_penalty`	the size of the penalty for all parameters; default is 0 (no penalty)

`is_complete_tree`
: whether or not a tree with all its extinct species is provided

`func`
: function to be used in solving the ODE system. Currently only for testing purposes.

`verbose`
: sets verbose output; default is verbose when optimmethod is 'submplex'

Value

Parameter estimated and maximum likelihood

Note

To run in parallel it is needed to load the following libraries when windows: apTreeshape, doparallel and foreach. When unix, it requires: apTreeshape, doparallel, foreach and doMC

Examples

```
# Example of how to set the arguments for a ML search.
library(secsse)
library(DDD)
set.seed(13)
# Check the vignette for a better working exercise.
# lambdas for 0A and 1A and 2A are the same but need to be estimated (CTD model, see Syst Biol
# paper)
# mus are fixed to zero,
# the transition rates are constrained to be equal and fixed 0.01
phylotree <- ape::rcoal(31, tip.label = 1:31)
traits <-  sample(c(0,1,2), ape::Ntip(phylotree),replace=TRUE) #get some traits
num_concealed_states <- 3
idparslist <- cla_id_paramPos(traits,num_concealed_states)
idparslist$lambdas[1,] <- c(1,1,1,2,2,2,3,3,3)
idparslist[[2]][] <- 4
masterBlock <- matrix(5,ncol = 3,nrow = 3,byrow = TRUE)
diag(masterBlock) <- NA
diff.conceal <- FALSE
idparslist[[3]] <- q_doubletrans(traits,masterBlock,diff.conceal)
startingpoint <- bd_ML(brts = ape::branching.times(phylotree))
intGuessLamba <- startingpoint$lambda0
intGuessMu <- startingpoint$mu0
idparsopt <- c(1,2,3)
initparsopt <- c(rep(intGuessLamba,3))
idparsfix <- c(0,4,5)
parsfix <- c(0,0,0.01)
tol <- c(1e-04, 1e-05, 1e-07)
maxiter <- 1000 * round((1.25) ^ length(idparsopt))
use_fortran <- FALSE
methode <- "ode45"
optimmethod <- "simplex"
run_parallel <- FALSE
cond <- "proper_cond"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
```

```
#model <- cla_secsse_ml(
#  phylotree,
#  traits,
#  num_concealed_states,
#  idparslist,
#  idparsopt,
#  initparsopt,
#  idparsfix,
#  parsfix,
#  cond,
#  root_state_weight,
#  sampling_fraction,
#  tol,
#  maxiter,
#  use_fortran,
#  methode,
#  optimmethod,
#  num_cycles = 1,
#  run_parallel)
# [1] -90.97626
```

`cla_secsse_ml_func_def_pars` *Maximum likehood estimation for (SecSSE) with parameter as complex functions. Cladogenetic version*

Description

Maximum likehood estimation under cla Several examined and concealed States-dependent Speciation and Extinction (SecSSE) where some paramaters are functions of other parameters and/or factors. Offers the option of cladogenesis

Usage

```
cla_secsse_ml_func_def_pars(
  phy,
  traits,
  num_concealed_states,
  idparslist,
  idparsopt,
  initparsopt,
  idfactorsopt,
  initfactors,
  idparsfix,
  parsfix,
  idparsfuncdefpar,
  functions_defining_params,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  tol = c(1e-04, 1e-05, 1e-07),
  maxiter = 1000 * round((1.25)^length(idparsopt)),
  use_fortran = TRUE,
  methode = "ode45",
  optimmethod = "simplex",
  num_cycles = 1,
  run_parallel = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  func = "cla_secsse_runmod",
  verbose = (optimmethod == "subplex")
)
```

Arguments

`phy`	phylogenetic tree of class phylo, ultrametric, rooted and with branch lengths.
`traits`	a vector with trait states for each tip in the phylogeny.
`num_concealed_states`	number of concealed states, generally equivalent to the number of examined states in the dataset.
`idparslist`	overview of parameters and their values.
`idparsopt`	id of parameters to be estimated.
`initparsopt`	initial guess of the parameters to be estimated.
`idfactorsopt`	id of the factors that will be optimized. There are not fixed factors, so use a constant within 'functions_defining_params'.
`initfactors`	the initial guess for a factor (it should be set to NULL when no factors).
`idparsfix`	id of the fixed parameters (it should be set to NULL when no factors).
`parsfix`	value of the fixed parameters.
`idparsfuncdefpar`	id of the parameters which will be a function of optimized and/or fixed parameters. The order of id should match functions_defining_params
`functions_defining_params`	a list of functions. Each element will be a function which defines a parameter e.g. id_3 <- (id_1+id_2)/2. See example and vigenette
`cond`	condition on the existence of a node root: "maddison_cond","proper_cond"(default). For details, see vignette.
`root_state_weight`	the method to weigh the states:"maddison_weights","proper_weights"(default) or "equal_weights". It can also be specified the root state:the vector c(1,0,0) indicates state 1 was the root state.
`sampling_fraction`	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states.

`tol`	maximum tolerance. Default is "c(1e-04, 1e-05, 1e-05)".
`maxiter`	max number of iterations. Default is "1000 *round((1.25)^length(idparsopt))".
`use_fortran`	Should the Fortran code for numerical integration be called? Default is TRUE.
`methode`	method used for integration calculation. Default is "ode45".
`optimmethod`	method used for optimization. Default is "simplex".
`num_cycles`	number of cycles of the optimization (default is 1).
`run_parallel`	should the routine to run in parallel be called? Read note below
`loglik_penalty`	the size of the penalty for all parameters; default is 0 (no penalty)
`is_complete_tree`	whether or not a tree with all its extinct species is provided
`func`	function to be used in solving the ODE system. Currently only for testing purposes.
`verbose`	sets verbose output; default is verbose when optimmethod is 'subplex'

Value

Parameter estimated and maximum likelihood

Note

To run in parallel the following libraries must be loaded under windows: apTreeshape, doparallel and foreach. Under unix/linux, apTreeshape, doparallel, foreach and doMC must be loaded.

Examples

```
# Example of how to set the arguments for a ML search.
rm(list=ls(all=TRUE))
library(secsse)
library(DDD)
set.seed(16)
phylotree <- ape::rbdtree(0.07,0.001,Tmax=50)
startingpoint <- bd_ML(brts = ape::branching.times(phylotree))
intGuessLamba <- startingpoint$lambda0
intGuessMu <- startingpoint$mu0
traits <-  sample(c(0,1,2), ape::Ntip(phylotree),replace=TRUE) #get some traits
num_concealed_states <- 3
idparslist <- cla_id_paramPos(traits,num_concealed_states)
idparslist$lambdas[1,] <- c(1,2,3,1,2,3,1,2,3)
idparslist[[2]][] <- 4
masterBlock <- matrix(c(5,6,5,6,5,6,5,6,5),ncol=3,nrow=3,byrow=TRUE)
diag(masterBlock) <- NA
diff.conceal <- FALSE
idparslist[[3]] <- q_doubletrans(traits,masterBlock,diff.conceal)
idparsfuncdefpar <- c(3,5,6)
idparsopt <- c(1,2)
idparsfix <- c(0,4)
initparsopt <- c(rep(intGuessLamba,2))
parsfix <- c(0,0)
```

```
idfactorsopt <- 1
initfactors <- 4
# functions_defining_params is a list of functions. Each function has no arguments and to refer
# to parameters ids should be indicated as "par_" i.e. par_3 refers to parameter 3. When a
# function is defined, be sure that all the parameters involved are either estimated, fixed or
# defined by previous functions (i.e, a function that defines parameter in
# 'functions_defining_params'). The user is responsible for this. In this example, par_3
# (i.e., parameter 3) is needed to calculate par_6. This is correct because par_3 is defined
# in the first function of 'functions_defining_params'. Notice that factor_1 indicates a value
# that will be estimated to satisfy the equation. The same factor can be shared to define
# several parameters.
functions_defining_params <- list()
functions_defining_params[[1]] <- function(){
 par_3 <- par_1 + par_2
}
functions_defining_params[[2]] <- function(){
 par_5 <- par_1 * factor_1
}
functions_defining_params[[3]] <- function(){
 par_6 <- par_3 * factor_1
}

tol = c(1e-04, 1e-05, 1e-07)
maxiter = 1000 * round((1.25)^length(idparsopt))
use_fortran = TRUE
methode = "ode45"
optimmethod = "simplex"
run_parallel = FALSE
cond <- "proper_cond"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
#model <- cla_secsse_ml_func_def_pars(phylotree, traits, num_concealed_states, idparslist,
#                                    idparsopt, initparsopt, idfactorsopt, initfactors,
#                                    idparsfix, parsfix, idparsfuncdefpar,
#                                    functions_defining_params, cond, root_state_weight,
#                                    sampling_fraction, tol, maxiter, use_fortran,
#                                    methode, optimmethod, num_cycles = 1,run_parallel)

# ML -136.5796
```

`event_times` *Event times of a (possibly non-ultrametric) phylogenetic tree*

Description

Times at which speciation or extinction occurs

Usage

```
event_times(phy)
```

Arguments

`phy`	phylogenetic tree of class phylo, without polytomies, rooted and with branch lengths. Need not be ultrametric.

Value

times at which speciation or extinction happens.

Note

This script has been modified from BAMMtools' internal function NU.branching.times

---

`example_phy_GeoSSE` *A phylogeny with traits at the tips*

Description

An example phylogeny for testing purposes

Usage

```
phy
```

Format

A phylogeny as created by GeoSSE (diversitree)

`id_paramPos` *Parameter structure setting*

Description

It sets the parameters (speciation, extinction and transition) ids. Needed for ML calculation (secsse_ml)

Usage

```
id_paramPos(traits, num_concealed_states)
```

Arguments

`traits`	vector with trait states, order of states must be the same as tree tips, for help, see vignette.
`num_concealed_states`	number of concealed states, generally equivalent to number of examined states.

Value

A list that includes the ids of the parameters for ML analysis.

Examples

```
traits <- sample(c(0,1,2), 45,replace = TRUE) #get some traits
num_concealed_states <- 3
param_posit <- id_paramPos(traits,num_concealed_states)
```

---

phylo_Vign *A phylogenetic reconstuction to run the vignette*

Description

An example phylogeny in the right format for secsse

Format

Phylogenetic tree in format nexus, rooted, including branch lengths

---

prepare_full_lambdas *Prepares the entire set of lambda matrices for cla_secsse.*

Description

It provides the set of matrices containing all the speciation rates

Usage

```
prepare_full_lambdas(traits, num_concealed_states, lambd_and_modeSpe)
```

Arguments

`traits`	vector with trait states, order of states must be the same as tree tips, for help, see vignette.
`num_concealed_states`	number of concealed states, generally equivalent to number of examined states.
`lambd_and_modeSpe`	a matrix with the 4 models of speciation possible.

Value

A list of lambdas, its length would be the same than the number of trait states * num_concealed_states..

`q_doubletrans` *Basic Qmatrix*

Description

Sets a Q matrix where double transitions are not allowed

Usage

```
q_doubletrans(traits, masterBlock, diff.conceal)
```

Arguments

`traits`	vector with trait states, order of states must be the same as tree tips, for help, see vignette.
`masterBlock`	matrix of transitions among only examined states, NA in the main diagonal, used to build the full transition rates matrix.
`diff.conceal`	should the concealed states be different? Normally it should be FALSE.

Value

Q matrix that includes both examined and concealed states, it should be declared as the third element of idparslist.

Examples

```
traits <- sample(c(0,1,2), 45,replace = TRUE) #get some traits
masterBlock <- matrix(99,ncol = 3,nrow = 3,byrow = TRUE) #For a three-state trait
diag(masterBlock) <- NA
masterBlock[1,2] <- 6
masterBlock[1,3] <- 7
masterBlock[2,1] <- 8
masterBlock[2,3] <- 9
masterBlock[3,1] <- 10
masterBlock[3,2] <- 11
myQ <- q_doubletrans(traits,masterBlock,diff.conceal = FALSE)
# now, it can replace the Q matrix from id_paramPos
num_concealed_states <- 3
param_posit <- id_paramPos(traits,num_concealed_states)
param_posit[[3]] <- myQ
```

secsse_loglik *Likelihood for SecSSE model*

Description

Logikelihood calculation for the SecSSE model given a set of parameters and data

Usage

```
secsse_loglik(
  parameter,
  phy,
  traits,
  num_concealed_states,
  use_fortran = TRUE,
  methode = "ode45",
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  run_parallel = FALSE,
  setting_calculation = NULL,
  setting_parallel = NULL,
  see_ancestral_states = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  func = ifelse(is_complete_tree, "secsse_runmod_ct", ifelse(use_fortran == FALSE,
    secsse_loglik_rhs, "secsse_runmod2"))
)
```

Arguments

`parameter`	list where first vector represents lambdas, the second mus and the third transition rates.
`phy`	phylogenetic tree of class phylo, ultrametric, fully-resolved, rooted and with branch lengths.
`traits`	vector with trait states, order of states must be the same as tree tips, for help, see vignette.
`num_concealed_states`	number of concealed states, generally equivalent to number of examined states.
`use_fortran`	Should the Fortran code for numerical integration be called? Default is TRUE.
`methode`	Solver for the set of equations, default is "ode45".
`cond`	condition on the existence of a node root: "maddison_cond","proper_cond"(default). For details, see vignette.

`root_state_weight`	the method to weigh the states:"maddison_weights","proper_weights"(default) or "equal_weights". It can also be specified the root state:the vector c(1,0,0) indicates state 1 was the root state.
`sampling_fraction`	vector that states the sampling proportion per trait state. It must have as many elements as trait states.
`run_parallel`	should the routine to run in parallel be called?
`setting_calculation`	argument used internally to speed up calculation. It should be left blank (default : setting_calculation = NULL)
`setting_parallel`	argument used internally to set a parallel calculation. It should be left blank (default : setting_parallel = NULL)
`see_ancestral_states`	should the ancestral states be shown? Deafault FALSE
`loglik_penalty`	the size of the penalty for all parameters; default is 0 (no penalty)
`is_complete_tree`	whether or not a tree with all its extinct species is provided
`func`	function to be used in solving the ODE system Currently only for testing purposes.

Value

The loglikelihood of the data given the parameters

Note

To run in parallel it is needed to load the following libraries when windows: apTreeshape, doparallel and foreach. When unix, it requires: apTreeshape, doparallel, foreach and doMC

Examples

```
rm(list = ls(all = TRUE))
library(secsse)
library(DDD)
library(deSolve)
library(apTreeshape)
library(foreach)
set.seed(13)
phylotree <- ape::rcoal(31, tip.label = 1:31)
traits <- sample(c(0,1,2),ape::Ntip(phylotree),replace = TRUE)
num_concealed_states <- 2
use_fortran <- TRUE
cond <- "proper_cond"
methode <- "ode45"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
run_parallel <- FALSE
```

```
drill <- id_paramPos(traits,num_concealed_states)
drill[[1]][] <- c(0.12,0.01,0.2,0.21,0.31,0.23)
drill[[2]][] <- 0
drill[[3]][,] <- 0.1
diag(drill[[3]]) <- NA
secsse_loglik(parameter = drill,phylotree,traits,num_concealed_states,
  use_fortran,methode,cond,root_state_weight,sampling_fraction,see_ancestral_states = FALSE)

#[1] -113.1018
```

---

secsse_ml *Maximum likehood estimation for (SecSSE)*

Description

Maximum likehood estimation under Several examined and concealed States-dependent Speciation and Extinction (SecSSE)

Usage

```
secsse_ml(
  phy,
  traits,
  num_concealed_states,
  idparslist,
  idparsopt,
  initparsopt,
  idparsfix,
  parsfix,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  tol = c(1e-04, 1e-05, 1e-07),
  maxiter = 1000 * round((1.25)^length(idparsopt)),
  use_fortran = TRUE,
  methode = "ode45",
  optimmethod = "simplex",
  num_cycles = 1,
  run_parallel = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
 func = ifelse(is_complete_tree, "secsse_runmod_ct", ifelse(use_fortran == FALSE,
    secsse_loglik_rhs, "secsse_runmod2")),
  verbose = (optimmethod == "subplex")
)
```

Arguments

`phy`	phylogenetic tree of class phylo, ultrametric, rooted and with branch lengths.
`traits`	a vector with trait states for each tip in the phylogeny.
`num_concealed_states`	number of concealed states, generally equivalent to the number of examined states in the dataset.
`idparslist`	overview of parameters and their values.
`idparsopt`	id of parameters to be estimated.
`initparsopt`	initial guess of the parameters to be estimated.
`idparsfix`	id of the fixed parameters.
`parsfix`	value of the fixed parameters.
`cond`	condition on the existence of a node root: "maddison_cond","proper_cond"(default). For details, see vignette.
`root_state_weight`	the method to weigh the states:"maddison_weights","proper_weights"(default) or "equal_weights". It can also be specified the root state:the vector c(1,0,0) indicates state 1 was the root state.
`sampling_fraction`	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states.
`tol`	maximum tolerance. Default is "c(1e-04, 1e-05, 1e-05)".
`maxiter`	max number of iterations. Default is "1000 *round((1.25)^length(idparsopt))".
`use_fortran`	Should the Fortran code for numerical integration be called? Default is TRUE.
`methode`	method used for integration calculation. Default is "ode45".
`optimmethod`	method used for optimization. Default is "simplex".
`num_cycles`	number of cycles of the optimization (default is 1).
`run_parallel`	should the routine to run in parallel be called?
`loglik_penalty`	the size of the penalty for all parameters; default is 0 (no penalty)
`is_complete_tree`	whether or not a tree with all its extinct species is provided
`func`	function to be used in solving the ODE system. Currently only for testing purposes.
`verbose`	sets verbose output; default is verbose when optimmethod is 'subplex'

Value

Parameter estimated and maximum likelihood

Note

To run in parallel it is needed to load the following libraries when windows: apTreeshape, doparallel and foreach. When unix, it requires: apTreeshape, doparallel, foreach and doMC

Examples

```
# Example of how to set the arguments for a ML search.
library(secsse)
library(DDD)
set.seed(13)
# Check the vignette for a better working exercise.
# lambdas for 0A and 1A and 2A are the same but need to be estimated
# mus are fixed to
# the transition rates are constrained to be equal and fixed 0.01
phylotree <- ape::rcoal(31, tip.label = 1:31)
traits <-  sample(c(0,1,2), ape::Ntip(phylotree),replace=TRUE) #get some traits
num_concealed_states<-3
idparslist <- id_paramPos(traits, num_concealed_states)
idparslist[[1]][c(1,4,7)] <- 1
idparslist[[1]][c(2,5,8)] <- 2
idparslist[[1]][c(3,6,9)] <- 3
idparslist[[2]][]<-4
masterBlock <- matrix(5,ncol = 3,nrow = 3,byrow = TRUE)
diag(masterBlock) <- NA
diff.conceal <- FALSE
idparslist[[3]] <- q_doubletrans(traits,masterBlock,diff.conceal)
startingpoint <- bd_ML(brts = ape::branching.times(phylotree))
intGuessLamba <- startingpoint$lambda0
intGuessMu <- startingpoint$mu0
idparsopt <- c(1,2,3,5)
initparsopt <- c(rep(intGuessLamba,3),rep((intGuessLamba/5),1))
idparsfix <- c(0,4)
parsfix <- c(0,0)
tol <- c(1e-04, 1e-05, 1e-07)
maxiter <- 1000 * round((1.25)^length(idparsopt))
use_fortran <- TRUE
methode <- "ode45"
optimmethod <- "simplex"
run_parallel <- FALSE
cond <- "proper_cond"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
#model<-secsse_ml(
#phylotree,
#traits,
#num_concealed_states,
#idparslist,
#idparsopt,
#initparsopt,
#idparsfix,
#parsfix,
#cond,
#root_state_weight,
#sampling_fraction,
#tol,
#maxiter,
#use_fortran,
#methode,
#optimmethod,
#num_cycles = 1,
#run_parallel
#)
# $ML
# [1] -16.43162
```

---

secsse_ml_func_def_pars

Maximum likehood estimation for (SecSSE) with parameter as complex functions.

Description

Maximum likehood estimation under Several examined and concealed States-dependent Speciation and Extinction (SecSSE) where some paramaters are functions of other parameters and/or factors.

Usage

```
secsse_ml_func_def_pars(
  phy,
  traits,
  num_concealed_states,
  idparslist,
  idparsopt,
  initparsopt,
  idfactorsopt,
  initfactors,
  idparsfix,
  parsfix,
  idparsfuncdefpar,
  functions_defining_params,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  tol = c(1e-04, 1e-05, 1e-07),
  maxiter = 1000 * round((1.25)^length(idparsopt)),
  use_fortran = TRUE,
  methode = "ode45",
  optimmethod = "simplex",
  num_cycles = 1,
  run_parallel = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
 func = ifelse(is_complete_tree, "secsse_runmod_ct", ifelse(use_fortran == FALSE,
   secsse_loglik_rhs, "secsse_runmod2"))
)
```

Arguments

`phy`	phylogenetic tree of class phylo, ultrametric, rooted and with branch lengths.
`traits`	a vector with trait states for each tip in the phylogeny.
`num_concealed_states`	number of concealed states, generally equivalent to the number of examined states in the dataset.
`idparslist`	overview of parameters and their values.
`idparsopt`	id of parameters to be estimated.
`initparsopt`	initial guess of the parameters to be estimated.
`idfactorsopt`	id of the factors that will be optimized. There are not fixed factors, so use a constant within 'functions_defining_params'.
`initfactors`	the initial guess for a factor (it should be set to NULL when no factors).
`idparsfix`	id of the fixed parameters (it should be set to NULL when there are no factors).
`parsfix`	value of the fixed parameters.
`idparsfuncdefpar`	id of the parameters which will be a function of optimized and/or fixed parameters. The order of id should match functions_defining_params
`functions_defining_params`	a list of functions. Each element will be a function which defines a parameter e.g. id_3 <- (id_1+id_2)/2. See example and vigenette
`cond`	condition on the existence of a node root: "maddison_cond","proper_cond"(default). For details, see vignette.
`root_state_weight`	the method to weigh the states:"maddison_weights","proper_weights"(default) or "equal_weights". It can also be specified the root state:the vector c(1,0,0) indicates state 1 was the root state.
`sampling_fraction`	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states.
`tol`	maximum tolerance. Default is "c(1e-04, 1e-05, 1e-05)".
`maxiter`	max number of iterations. Default is "1000 *round((1.25)^length(idparsopt))".
`use_fortran`	Should the Fortran code for numerical integration be called? Default is TRUE.
`methode`	method used for integration calculation. Default is "ode45".
`optimmethod`	method used for optimization. Default is "simplex".
`num_cycles`	number of cycles of the optimization (default is 1).
`run_parallel`	should the routine to run in parallel be called? Read note below
`loglik_penalty`	the size of the penalty for all parameters; default is 0 (no penalty)
`is_complete_tree`	whether or not a tree with all its extinct species is provided
`func`	function to be used in solving the ODE system. Currently only for testing purposes.

Value

Parameter estimated and maximum likelihood

Note

To run in parallel it is needed to load the following libraries when windows: apTreeshape, doparallel and foreach. When unix, it requires: apTreeshape, doparallel, foreach and doMC

Examples

```
# Example of how to set the arguments for a ML search.
rm(list=ls(all=TRUE))
library(secsse)
library(DDD)
set.seed(16)
phylotree <- ape::rbdtree(0.07,0.001,Tmax=50)
startingpoint<-bd_ML(brts = ape::branching.times(phylotree))
intGuessLamba <- startingpoint$lambda0
intGuessMu <- startingpoint$mu0
traits <- sample(c(0,1,2), ape::Ntip(phylotree),replace=TRUE) #get some traits
num_concealed_states<-3
idparslist<-id_paramPos(traits, num_concealed_states)
idparslist[[1]][c(1,4,7)] <- 1
idparslist[[1]][c(2,5,8)] <- 2
idparslist[[1]][c(3,6,9)] <- 3
idparslist[[2]][]<-4
masterBlock <- matrix(c(5,6),ncol = 3,nrow = 3,byrow = TRUE)
diag(masterBlock) <- NA
diff.conceal <- FALSE
idparslist[[3]] <- q_doubletrans(traits,masterBlock,diff.conceal)
idparsfuncdefpar <- c(3,5,6)
idparsopt <- c(1,2)
idparsfix <- c(0,4)
initparsopt <- c(rep(intGuessLamba,2))
parsfix <- c(0,0)
idfactorsopt <- 1
initfactors <- 4
# functions_defining_params is a list of functions. Each function has no arguments and to refer
# to parameters ids should be indicated as "par_" i.e. par_3 refers to parameter 3. When a
# function is defined, be sure that all the parameters involved are either estimated, fixed or
# defined by previous functions (i.e, a function that defines parameter in
# 'functions_defining_params'). The user is responsible for this. In this example, par_3
# (i.e., parameter 3) is needed to calculate par_6. This is correct because par_3 is defined in
# the first function of 'functions_defining_params'. Notice that factor_1 indicates a value
# that will be estimated to satisfy the equation. The same factor can be shared to define
# several parameters.
functions_defining_params <- list()
functions_defining_params[[1]] <- function(){
 par_3 <- par_1 + par_2
}
functions_defining_params[[2]] <- function(){
 par_5 <- par_1 * factor_1
}
functions_defining_params[[3]] <- function(){
 par_6 <- par_3 * factor_1
}

tol = c(1e-04, 1e-05, 1e-07)
maxiter = 1000 * round((1.25)^length(idparsopt))
use_fortran = TRUE
methode = "ode45"
optimmethod = "simplex"
run_parallel = FALSE
cond<-"proper_cond"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
#model <- secsse_ml_func_def_pars(phylotree, traits, num_concealed_states, idparslist, idparsopt,
#                              initparsopt, idfactorsopt, initfactors, idparsfix, parsfix,
#                                 idparsfuncdefpar, functions_defining_params, cond,
#                         root_state_weight, sampling_fraction, tol, maxiter, use_fortran,
#                                 methode, optimmethod, num_cycles = 1, run_parallel)

# ML -136.5796
```

`sortingtraits` *Data checking and trait sorting*

Description

In preparation for likelihood calculation, it orders trait data according the tree tips

Usage

```
sortingtraits(traitinfo, phy)
```

Arguments

traitinfo	data frame where first column has species ids and the second one is the trait associated information.
phy	phy phylogenetic tree of class phylo, ultrametric, fully-resolved, rooted and with branch lengths.

Value

Vector of traits

Examples

```
# Some data we have prepared
data(traitinfo)
data("phylo_Vign")
traits <- sortingtraits(traitinfo,phylo_Vign)
```

`traitinfo` *A table with trait info to run the vignette*

Description

An example of trait information in the right format for secsse

Format

A data frame where each species has a trait state associated

Index