

# Package ‘secsse’

July 24, 2025

**Type** Package

**Title** Several Examined and Concealed States-Dependent Speciation and Extinction

**Version** 3.5.0

**Date** 2025-07-23

**License** GPL (>= 3) | file LICENSE

**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) [doi:10.1093/sysbio/syy057](https://doi.org/10.1093/sysbio/syy057).

**Depends** R (>= 4.2.0)

**Imports** utils, DDD (>= 5.0), ape, geiger, Rcpp (>= 1.0.10), RcppParallel, ggplot2, tibble, rlang, treestats

**Suggests** diversitree, phytools, testthat, subplex, knitr, rmarkdown

**LinkingTo** Rcpp, RcppParallel, BH (>= 1.81.0-1)

**NeedsCompilation** yes

**SystemRequirements** C++17

**Encoding** UTF-8

**LazyData** true

**URL** <https://rsetienne.github.io/secsse/>,  
<https://github.com/rsetienne/secsse>

**BugReports** <https://github.com/rsetienne/secsse/issues>

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**Author** Leonel Herrera Alsina [aut] (ORCID:  
<https://orcid.org/0000-0003-0474-3592>),  
Paul van Els [aut] (ORCID: <https://orcid.org/0000-0002-9499-8873>),  
Thijs Janzen [ctb] (ORCID: <https://orcid.org/0000-0002-4162-1140>),  
Hanno Hildenbrandt [ctb] (ORCID:

<<https://orcid.org/0000-0002-6784-1037>>),  
 Pedro Santos Neves [ctb] (ORCID:  
 <<https://orcid.org/0000-0003-2561-4677>>),  
 Rampal S. Etienne [cre, aut] (ORCID:  
 <<https://orcid.org/0000-0003-2142-7612>>)

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

**Repository** CRAN

**Date/Publication** 2025-07-24 16:10:07 UTC

## Contents

cla_id_paramPos . . . . .	3
cla_secsse_loglik . . . . .	3
cla_secsse_ml . . . . .	6
cla_secsse_ml_func_def_pars . . . . .	9
create_default_lambda_transition_matrix . . . . .	13
create_default_shift_matrix . . . . .	14
create_lambda_list . . . . .	15
create_mu_vector . . . . .	16
create_q_matrix . . . . .	17
event_times . . . . .	18
example_phy_GeoSSE . . . . .	18
expand_q_matrix . . . . .	19
extract_par_vals . . . . .	19
fill_in . . . . .	20
id_paramPos . . . . .	21
phylo_vignette . . . . .	21
plot_idparslist . . . . .	22
plot_state_exact . . . . .	23
prepare_full_lambdas . . . . .	25
q_doubletrans . . . . .	26
secsse_loglik . . . . .	27
secsse_loglik_eval . . . . .	30
secsse_ml . . . . .	32
secsse_ml_func_def_pars . . . . .	35
secsse_sim . . . . .	39
secsse_single_branch_loglik . . . . .	41
sortingtraits . . . . .	43
traits . . . . .	44

<b>Index</b>	<b>45</b>
--------------	-----------

---

cla_id_paramPos	<i>Parameter structure setting for cla_secsse It sets the parameters (speciation, extinction and transition) IDs. Needed for ML calculation with cladogenetic options (cla_secsse_ml)</i>
-----------------	---

---

### Description

Parameter structure setting for cla\_secsse 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 for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.

### 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	<i>Likelihood for SecSSE model, using Rcpp Loglikelihood calculation for the cla_SecSSE model given a set of parameters and data using Rcpp</i>
-------------------	---

---

### Description

Likelihood for SecSSE model, using Rcpp Loglikelihood calculation for the cla\_SecSSE model given a set of parameters and data using Rcpp

**Usage**

```
cla_secsse_loglik(
  parameter,
  phy,
  traits,
  num_concealed_states,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  setting_calculation = NULL,
  see_ancestral_states = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  take_into_account_root_edge = FALSE,
  num_threads = 1,
  method = "odeint::bulirsch_stoer",
  atol = 1e-08,
  rtol = 1e-07,
  display_warning = TRUE,
  use_normalization = TRUE
)
```

**Arguments**

parameter	list where first vector represents lambdas, the second mus and the third transition rates.
phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
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 for the root state: the vector c(1, 0, 0) indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling

	fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
setting_calculation	argument used internally to speed up calculation. It should be left blank (default : setting_calculation = NULL).
see_ancestral_states	Boolean for whether the ancestral states should be shown? Defaults to FALSE.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.
take_into_account_root_edge	if TRUE, the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)
num_threads	number of threads to be used. Default is one thread.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoe".
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
display_warning	display a warning if necessary
use_normalization	normalize the density vector during integration, more accurate but slower (default = TRUE)

## Value

The loglikelihood of the data given the parameters

## Examples

```
rm(list=ls(all=TRUE))
library(secsse)
set.seed(13)
phyloree <- ape::rcoal(12, tip.label = 1:12)
traits <- sample(c(0,1,2),ape::Ntip(phyloree),replace=TRUE)
num_concealed_states <- 3
sampling_fraction <- c(1,1,1)
phy <- phyloree
# 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
lambda_and_modeSpe <- parameter$lambdas
lambda_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,
lambda_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,
cond = 'maddison_cond',
root_state_weight = 'maddison_weights', sampling_fraction,
setting_calculation = NULL,
see_ancestral_states = FALSE,
loglik_penalty = 0)
# LL = -42.18407

```

---

cla\_secsse\_ml

---

*Maximum likelihood estimation for (SecSSE)*


---

## Description

Maximum likelihood 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)),
optimmethod = "simplex",
num_cycles = 1,
loglik_penalty = 0,
is_complete_tree = FALSE,
take_into_account_root_edge = FALSE,
verbose = FALSE,
num_threads = 1,
atol = 1e-08,
rtol = 1e-07,
method = "odeint::bulirsch_stoer",
use_normalization = TRUE
)

```

## Arguments

phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
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	a numeric vector with the ID of parameters to be estimated.
initparsopt	a numeric vector with the initial guess of the parameters to be estimated.
idparsfix	a numeric vector with the ID of the fixed parameters.
parsfix	a numeric vector with the 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 for the root state: the vector c(1, 0, 0) indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
tol	A numeric vector with the maximum tolerance of the optimization algorithm. Default is c(1e-04, 1e-05, 1e-05).

maxiter	max number of iterations. Default is $1000 * \text{round}((1.25) ^ \text{length}(\text{idparsopt}))$ .
optimmethod	A string with method used for optimization. Default is "simplex". Alternative is "subplex". Both are called from <code>DDD::optimizer()</code> , simplex is implemented natively in <b>DDD</b> , while subplex is ultimately called from <code>subplex::subplex()</code> .
num_cycles	Number of cycles of the optimization. When set to Inf, the optimization will be repeated until the result is, within the tolerance, equal to the starting values, with a maximum of 10 cycles.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.
take_into_account_root_edge	if TRUE, the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)
verbose	sets verbose output; default is TRUE when optimmethod is "simplex". If optimmethod is set to "simplex", then even if set to FALSE, optimizer output will be shown.
num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoer".
use_normalization	normalize the density vector during integration, more accurate but slower (default = TRUE)

## Value

Parameter estimated and maximum likelihood

## 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)
# get some traits
traits <- sample(c(0,1,2), ape::Ntip(phylotree), replace = TRUE)
num_concealed_states <- 3
```



```

idparslist <- cla_id_paramPos(traits,num_concealed_states)
idparslist$lambdas[1,] <- c(1,1,1,2,2,2,3,3,3)
idparslist[[2]][,] <- 4
masterBlock <- matrix(5,ncol = 3,nrow = 3,byrow = TRUE)
diag(masterBlock) <- NA
diff.conceal <- FALSE
idparslist[[3]] <- q_doubletrans(traits,masterBlock,diff.conceal)
startingpoint <- bd_ML(brts = ape::branching.times(phyloree))
intGuessLamba <- startingpoint$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-03, 1e-03, 1e-03)
maxiter <- 1000 * round((1.25) ^ length(idparsopt))
optimmethod <- 'simplex'
cond <- 'proper_cond'
root_state_weight <- 'proper_weights'
sampling_fraction <- c(1,1,1)
model <- cla_secsse_ml(
  phyloree,
  traits,
  num_concealed_states,
  idparslist,
  idparsopt,
  initparsopt,
  idparsfix,
  parsfix,
  cond,
  root_state_weight,
  sampling_fraction,
  tol,
  maxiter,
  optimmethod,
  num_cycles = 1,
  num_threads = 1,
  verbose = FALSE)
# [1] -90.9763

```

---

cla\_secsse\_ml\_func\_def\_pars

*Maximum likelihood estimation for (SecSSE) with parameter as complex functions. Cladogenetic version*

---

## Description

Maximum likelihood 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)),
  optimmethod = "simplex",
  num_cycles = 1,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  take_into_account_root_edge = FALSE,
  verbose = TRUE,
  num_threads = 1,
  atol = 1e-12,
  rtol = 1e-12,
  method = "odeint::bulirsch_stoer",
  use_normalization = TRUE
)
```

**Arguments**

phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
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	a numeric vector with the ID of parameters to be estimated.
initparsopt	a numeric vector with the 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	a numeric vector with the ID of the fixed parameters.
parsfix	a numeric vector with the 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. <code>id_3 &lt;- (id_1 + id_2) / 2</code> . See example.
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 for the root state: the vector <code>c(1, 0, 0)</code> indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
tol	A numeric vector with the maximum tolerance of the optimization algorithm. Default is <code>c(1e-04, 1e-05, 1e-05)</code> .
maxiter	max number of iterations. Default is <code>1000 * round((1.25) ^ length(idparsopt))</code> .
optimmethod	A string with method used for optimization. Default is "simplex". Alternative is "subplex". Both are called from <code>DDD::optimizer()</code> , simplex is implemented natively in <b>DDD</b> , while subplex is ultimately called from <code>subplex::subplex()</code> .
num_cycles	Number of cycles of the optimization. When set to Inf, the optimization will be repeated until the result is, within the tolerance, equal to the starting values, with a maximum of 10 cycles.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.
take_into_account_root_edge	if TRUE, the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)
verbose	sets verbose output; default is TRUE when optimmethod is "simplex". If optimmethod is set to "simplex", then even if set to FALSE, optimizer output will be shown.

num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoer".
use_normalization	normalize the density vector during integration, more accurate but slower (default = TRUE)

## Value

Parameter estimated and maximum likelihood

## Examples

```
# Example of how to set the arguments for an ML search. The ML search is
# stopped after 10 iterations to keep the run time short.
library(secsse)
library(DDD)
set.seed(16)
phylo.tree <- ape::rbdtree(0.07,0.001,Tmax=50)
startingpoint <- bd_ML(brts = ape::branching.times(phylo.tree))
intGuessLamba <- startingpoint$lambda0
intGuessMu <- startingpoint$mu0
traits <- sample(c(0,1,2),
                 ape::Ntip(phylo.tree), replace = TRUE) # get some traits
num_concealed_states <- 3
idparslist <- cla_id_paramPos(traits, num_concealed_states)
idparslist$lambda0 <- c(1,2,3,1,2,3,1,2,3)
idparslist[[2]][1] <- 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-02, 1e-03, 1e-04)
optimmethod = 'subplex'
cond <- 'proper_cond'
root_state_weight <- 'proper_weights'
sampling_fraction <- c(1,1,1)
maxiter <- 10
model <- cla_secsse_ml_func_def_pars(phylo tree,
  traits,
  num_concealed_states,
  idparslist,
  idparsopt,
  initparsopt,
  idfactorsopt,
  initfactors,
  idparsfix,
  parsfix,
  idparsfuncdefpar,
  functions_defining_params,
  cond,
  root_state_weight,
  sampling_fraction,
  tol,
  maxiter,
  optimmethod,
  num_cycles = 1)
# ML -136.5796 if run till completion

```

---

```
create_default_lambda_transition_matrix
```

*Helper function to create a default lambda list*

---

**Description**

This function generates a generic lambda list, assuming no transitions between states, e.g. a species of observed state 0 generates daughter species with state 0 as well.

**Usage**

```
create_default_lambda_transition_matrix(
  state_names = c("0", "1"),
  model = "ETD"
)
```

**Arguments**

**state\_names**      vector of names of all observed states.

**model**            used model, choice of "ETD" (Examined Traits Diversification), "CTD" (Concealed Traits Diversification) or "CR" (Constant Rate).

**Examples**

```
lambda_matrix <-
  create_default_lambda_transition_matrix(state_names = c(0, 1),
                                         model = "ETD")
lambda_list <- create_lambda_list(state_names = c(0, 1),
                                  num_concealed_states = 2,
                                  transition_matrix = lambda_matrix,
                                  model = "ETD")
```

---

```
create_default_shift_matrix
```

*Helper function to create a default shift\_matrix list*

---

**Description**

This function generates a generic shift matrix to be used with the function [create\\_q\\_matrix\(\)](#).

**Usage**

```
create_default_shift_matrix(
  state_names = c("0", "1"),
  num_concealed_states = 2,
  mu_vector = NULL
)
```

**Arguments**

state\_names      vector of names of all observed states.  
 num\_concealed\_states      number of concealed states, generally equivalent to the number of examined states in the dataset.  
 mu\_vector      previously defined mus - used to choose indicator number.

**Examples**

```
shift_matrix <- create_default_shift_matrix(state_names = c(0, 1),
                                           num_concealed_states = 2,
                                           mu_vector = c(1, 2, 1, 2))
q_matrix <- create_q_matrix(state_names = c(0, 1),
                           num_concealed_states = 2,
                           shift_matrix = shift_matrix,
                           diff.conceal = FALSE)
```

---

create_lambda_list	<i>Helper function to automatically create lambda matrices, based on input. When choosing the CTD model, rates associated with observed states are now re-distributed to concealed states. This implicitly assumes that the number of observed and concealed states is identical.</i>
--------------------	---

---

**Description**

Helper function to automatically create lambda matrices, based on input. When choosing the CTD model, rates associated with observed states are now re-distributed to concealed states. This implicitly assumes that the number of observed and concealed states is identical.

**Usage**

```
create_lambda_list(
  state_names = c(0, 1),
  num_concealed_states = 2,
  transition_matrix,
  model = "ETD"
)
```

**Arguments**

state\_names      vector of names of all observed states.  
 num\_concealed\_states      number of concealed states, generally equivalent to the number of examined states in the dataset.

transition_matrix	a matrix containing a description of all speciation events, where the first column indicates the source state, the second and third column indicate the two daughter states, and the fourth column gives the rate indicator used. E.g.: ["SA", "S", "A", 1] for a trait state "SA" which upon speciation generates two daughter species with traits "S" and "A", where the number 1 is used as indicator for optimization of the likelihood.
model	used model, choice of "ETD" (Examined Traits Diversification), "CTD" (Concealed Traits Diversification) or "CR" (Constant Rate).

### Examples

```
trans_matrix <- c(0, 0, 0, 1)
trans_matrix <- rbind(trans_matrix, c(1, 1, 1, 2))
lambda_list <- create_lambda_list(state_names = c(0, 1),
                                num_concealed_states = 2,
                                transition_matrix = trans_matrix,
                                model = "ETD")
```

---

create_mu_vector	<i>Generate mus vector</i>
------------------	----------------------------

---

### Description

Generate mus vector

### Usage

```
create_mu_vector(state_names, num_concealed_states, model = "CR", lambda_list)
```

### Arguments

state_names	vector of names of all observed states.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
model	used model, choice of "ETD" (Examined Traits Diversification), "CTD" (Concealed Traits Diversification) or "CR" (Constant Rate).
lambda_list	previously generated list of lambda matrices, used to infer the rate number to start with.

### Value

mu vector





---

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**

```
example_phy_GeoSSE
```

**Format**

A phylogeny as created by GeoSSE (diversitree)

---

expand_q_matrix	<i>Function to expand an existing q_matrix to a number of concealed states</i>
-----------------	--

---

**Description**

Function to expand an existing q\_matrix to a number of concealed states

**Usage**

```
expand_q_matrix(q_matrix, num_concealed_states, diff.conceal = FALSE)
```

**Arguments**

q_matrix	q_matrix with only transitions between observed states.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
diff.conceal	Boolean stating if the concealed states should be different. E.g. that the transition rates for the concealed states are different from the transition rates for the examined states. Normally it should be FALSE in order to avoid having a huge number of parameters.

**Value**

updated q matrix

**Note**

This is highly similar to [q\\_doubletrans\(\)](#).

---

extract_par_vals	<i>Extract parameter values out of the result of a maximum likelihood inference run</i>
------------------	---

---

**Description**

Extract parameter values out of the result of a maximum likelihood inference run

**Usage**

```
extract_par_vals(param_posit, ml_pars)
```

**Arguments**

param_posit	initial parameter structure, consisting of a list with three entries: <ol style="list-style-type: none"> <li>1. lambda matrices</li> <li>2. mus</li> <li>3. Q matrix</li> </ol> <p>In each entry, integers numbers (1-n) indicate the parameter to be optimized.</p>
ml_pars	resulting parameter estimates as returned by for instance <code>cla_secsse_ml()</code> , having the same structure as <code>param_post</code> .

**Value**

Vector of parameter estimates.

---

fill\_in

---

*Helper function to enter parameter value on their right place*


---

**Description**

Helper function to enter parameter value on their right place

**Usage**

```
fill_in(object, params)
```

**Arguments**

object	lambda matrices, <code>q_matrix</code> or mu vector.
params	parameters in order, where each value reflects the value of the parameter at that position, e.g. <code>c(0.3, 0.2, 0.1)</code> will fill out the value 0.3 for the parameter with rate identifier 1, 0.2 for the parameter with rate identifier 2 and 0.1 for the parameter with rate identifier 3.

**Value**

lambda matrices, `q_matrix` or mu vector with the correct values in their right place.

---

id_paramPos	<i>Parameter structure setting Sets the parameters (speciation, extinction and transition) ids. Needed for ML calculation (<a href="#">secsse_ml()</a>).</i>
-------------	--

---

### Description

Parameter structure setting 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 for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see <code>vignette("starting_secsse", package = "secsse")</code> . When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.

### 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_vignette	<i>A phylogenetic reconstruction to run the vignette</i>
----------------	--

---

### Description

An example phylogeny in the right format for secsse

### Usage

```
phylo_vignette
```

### Format

Phylogenetic tree in phy format, rooted, including branch lengths

---

plot_idparslist	<i>function to visualize the structure of the idparslist</i>
-----------------	--

---

**Description**

function to visualize the structure of the idparslist

**Usage**

```
plot_idparslist(idparslist, state_names)
```

**Arguments**

idparslist	idparslist setup list
state_names	names of all states (including concealed states)

**Value**

list with two ggplot objects: plot\_qmat which visualizes the q\_matrix, and plot\_lambda which visualizes the lambda matrices.

**Examples**

```
idparslist <- list()
focal_matrix <-
secsse::create_default_lambda_transition_matrix(state_names = c("1", "2"),
                                                model = "CR")

idparslist[[1]] <-
  secsse::create_lambda_list(state_names = c("1", "2"),
                            num_concealed_states = 2,
                            transition_matrix = focal_matrix,
                            model = "CR")

idparslist[[2]] <- secsse::create_mu_vector(state_names = c("1", "2"),
                                           num_concealed_states = 2,
                                           model = "CR",
                                           lambda_list = idparslist[[1]])

shift_mat <- secsse::create_default_shift_matrix(state_names = c("1", "2"),
                                                num_concealed_states = 2,
                                                mu_vector = idparslist[[2]])

idparslist[[3]] <- secsse::create_q_matrix(state_names = c("1", "2"),
                                          num_concealed_states = 2,
                                          shift_matrix = shift_mat,
                                          diff.conceal = FALSE)

p <- plot_idparslist(idparslist, state_names = names(idparslist[[1]]))
p$plot_lambda
p$plot_qmat
```

---

plot_state_exact	<i>Plot the local probability along a tree</i>
------------------	--

---

## Description

Plot the local probability along the tree, including the branches

## Usage

```
plot_state_exact(
  parameters,
  phy,
  traits,
  num_concealed_states,
  sampling_fraction,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  is_complete_tree = FALSE,
  method = "odeint::bulirsch_stoer",
  atol = 1e-16,
  rtol = 1e-16,
  num_steps = 100,
  prob_func = NULL,
  verbose = FALSE
)
```

## Arguments

parameters	list where first vector represents lambdas, the second mus and the third transition rates.
phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.

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 for the root state: the vector c(1, 0, 0) indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoe".
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
num_steps	number of substeps to show intermediate likelihoods along a branch.
prob_func	a function to calculate the probability of interest, see description.
verbose	sets verbose output; default is TRUE when optimmethod is "simplex". If optimmethod is set to "simplex", then even if set to FALSE, optimizer output will be shown.

## Details

This function will evaluate the log likelihood locally along all branches and plot the result. When num\_steps is left to NULL, all likelihood evaluations during integration are used for plotting. This may work for not too large trees, but may become very memory heavy for larger trees. Instead, the user can indicate a number of steps, which causes the probabilities to be evaluated at a distinct amount of steps along each branch (and the probabilities to be properly integrated in between these steps). This provides an approximation, but generally results look very similar to using the full evaluation. The function used for prob\_func will be highly dependent on your system. for instance, for a 3 observed, 2 hidden states model, the probability of state A is prob[1] + prob[2] + prob[3], normalized by the row sum. prob\_func will be applied to each row of the 'states' matrix (you can thus test your function on the states matrix returned when 'see\_ancestral\_states = TRUE'). Please note that the first N columns of the states matrix are the extinction rates, and the (N+1):2N columns belong to the speciation rates, where N = num\_obs\_states \* num\_concealed\_states. A typical prob\_func function will look like:

```
my_prob_func <- function(x) {
  return(sum(x[5:8]) / sum(x))
}
```

## Value

ggplot2 object



## Examples

```

set.seed(5)
phy <- ape::rphylo(n = 4, birth = 1, death = 0)
traits <- c(0, 1, 1, 0)
params <- secsse::id_paramPos(c(0, 1), 2)
params[[1]][ ] <- c(0.2, 0.2, 0.1, 0.1)
params[[2]][ ] <- 0.0
params[[3]][, ] <- 0.1
diag(params[[3]]) <- NA
# Thus, we have for both, rates
# 0A, 1A, 0B and 1B. If we are interested in the posterior probability of
# trait 0, we have to provide a helper function that sums the probabilities of
# 0A and 0B, e.g.:
helper_function <- function(x) {
  return(sum(x[c(5, 7)]) / sum(x)) # normalized by total sum, just in case.
}

out_plot <- plot_state_exact(parameters = params,
                             phy = phy,
                             traits = traits,
                             num_concealed_states = 2,
                             sampling_fraction = c(1, 1),
                             num_steps = 10,
                             prob_func = helper_function)

```

---

prepare_full_lambdas	<i>Prepares the entire set of lambda matrices for cla_secsse. It provides the set of matrices containing all the speciation rates</i>
----------------------	---

---

## Description

Prepares the entire set of lambda matrices for cla\_secsse. 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 for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
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..

**Examples**

```
set.seed(13)
phyloree <- ape::rcoal(12, tip.label = 1:12)
traits <- sample(c(0, 1, 2),
                ape::Ntip(phyloree), replace = TRUE)
num_concealed_states <- 3
# the idparlist for a ETD model (dual state inheritance model of evolution)
# would be set like this:
idparlist <- secsse::cla_id_paramPos(traits, num_concealed_states)
lambda_and_modeSpe <- idparlist$lambdas
lambda_and_modeSpe[1, ] <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
idparlist[[1]] <- lambda_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, clascse sorts the lambda matrices, so they look like
# a list with 9 matrices, corresponding to the 9 states
# (0A,1A,2A,0B, etc)

parameter <- idparlist
lambda_and_modeSpe <- parameter$lambdas
lambda_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,
                                       lambda_and_modeSpe)
```

---

q_doubletrans	<i>Basic Qmatrix Sets a Q matrix where double transitions are not allowed</i>
---------------	---

---

**Description**

This function expands the Q\_matrix, but it does so assuming that the number of concealed traits is equal to the number of examined traits, if you have a different number, you should consider looking at the function [expand\\_q\\_matrix\(\)](#).

**Usage**

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

**Arguments**

traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see <code>vignette("starting_secsse")</code> ,
--------	---

	package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
masterBlock	matrix of transitions among only examined states, NA in the main diagonal, used to build the full transition rates matrix.
diff.conceal	Boolean stating if the concealed states should be different. E.g. that the transition rates for the concealed states are different from the transition rates for the examined states. Normally it should be FALSE in order to avoid having a huge number of parameters.

### 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
# For a three-state trait
masterBlock <- matrix(99,ncol = 3,nrow = 3,byrow = TRUE)
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 Loglikelihood calculation for the SecSSE model given a set of parameters and data*

---

### Description

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

### Usage

```
secsse_loglik(
  parameter,
  phy,
  traits,
```

```

num_concealed_states,
cond = "proper_cond",
root_state_weight = "proper_weights",
sampling_fraction,
setting_calculation = NULL,
see_ancestral_states = FALSE,
loglik_penalty = 0,
is_complete_tree = FALSE,
take_into_account_root_edge = FALSE,
num_threads = 1,
atol = 1e-08,
rtol = 1e-07,
method = "odeint::bulirsch_stoer",
display_warning = TRUE,
use_normalization = TRUE
)

```

## Arguments

parameter	list where first vector represents lambdas, the second mus and the third transition rates.
phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
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 for the root state: the vector c(1, 0, 0) indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
setting_calculation	argument used internally to speed up calculation. It should be left blank (default : setting_calculation = NULL).

see_ancestral_states	Boolean for whether the ancestral states should be shown? Defaults to FALSE.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.
take_into_account_root_edge	if TRUE, the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)
num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoer".
display_warning	display a warning if necessary
use_normalization	normalize the density vector during integration, more accurate but slower (default = TRUE)

## Value

The loglikelihood of the data given the parameter.

## Examples

```
rm(list = ls(all = TRUE))
library(secsse)
set.seed(13)
phyloree <- ape::rcoal(31, tip.label = 1:31)
traits <- sample(c(0,1,2),ape::Ntip(phyloree),replace = TRUE)
num_concealed_states <- 2
cond <- "proper_cond"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
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,
  phyloree,
  traits,
  num_concealed_states,
  cond,
  root_state_weight,
```

```
sampling_fraction,
see_ancestral_states = FALSE)

#[1] -113.1018
```

---

secsse_loglik_eval	<i>Likelihood for SecSSE model Loglikelihood calculation for the SecSSE model given a set of parameters and data, returning also the likelihoods along the branches</i>
--------------------	---

---

## Description

Likelihood for SecSSE model Loglikelihood calculation for the SecSSE model given a set of parameters and data, returning also the likelihoods along the branches

## Usage

```
secsse_loglik_eval(
  parameter,
  phy,
  traits,
  num_concealed_states,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  setting_calculation = NULL,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  num_threads = 1,
  atol = 1e-08,
  rtol = 1e-07,
  method = "odeint::bulirsch_stoer",
  num_steps = 100
)
```

## Arguments

parameter	list where first vector represents lambdas, the second mus and the third transition rates.
phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see <code>vignette("starting_secsse", package = "secsse")</code> . When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.

num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
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 for the root state: the vector <code>c(1, 0, 0)</code> indicates state 1 was the root state. When using a <code>multiPhylo</code> object, <code>root_state_weight</code> should be list where each entry in the list corresponds to the <code>root_state_weight</code> for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a <code>multiPhylo</code> object, <code>sampling_fraction</code> should be list where each entry in the list corresponds to the sampling proportion for each tree.
setting_calculation	argument used internally to speed up calculation. It should be left blank (default : <code>setting_calculation = NULL</code> ).
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to <code>TRUE</code> , it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to <code>FALSE</code> .
num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoe".
num_steps	number of substeps to show intermediate likelihoods along a branch.

### Value

A list containing: "output", observed states along evaluated time points along all branches, used for plotting. "states" all ancestral states on the nodes and "duration", indicating the time taken for the total evaluation

### Examples

```
set.seed(5)
phy <- ape::rphylo(n = 4, birth = 1, death = 0)
traits <- c(0, 1, 1, 0)
params <- secsse::id_paramPos(c(0, 1), 2)
params[[1]][ ] <- c(0.2, 0.2, 0.1, 0.1)
params[[2]][ ] <- 0.0
params[[3]][, ] <- 0.1
```

```
diag(params[[3]]) <- NA

secsse_loglik_eval(parameter = params,
                   phy = phy,
                   traits = traits,
                   num_concealed_states = 2,
                   sampling_fraction = c(1, 1),
                   num_steps = 10)
```

---

secsse\_ml

*Maximum likelihood estimation for (SecSSE)*


---

## Description

Maximum likelihood 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)),
  optimmethod = "simplex",
  num_cycles = 1,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  take_into_account_root_edge = FALSE,
  verbose = FALSE,
  num_threads = 1,
  atol = 1e-08,
  rtol = 1e-07,
  method = "odeint::bulirsch_stoer",
  use_normalization = TRUE
)
```



**Arguments**

phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see <code>vignette("starting_secsse", package = "secsse")</code> . When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
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	a numeric vector with the ID of parameters to be estimated.
initparsopt	a numeric vector with the initial guess of the parameters to be estimated.
idparsfix	a numeric vector with the ID of the fixed parameters.
parsfix	a numeric vector with the 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 for the root state: the vector <code>c(1, 0, 0)</code> indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
tol	A numeric vector with the maximum tolerance of the optimization algorithm. Default is <code>c(1e-04, 1e-05, 1e-05)</code> .
maxiter	max number of iterations. Default is <code>1000 * round((1.25) ^ length(idparsopt))</code> .
optimmethod	A string with method used for optimization. Default is "simplex". Alternative is "subplex". Both are called from <code>DDD::optimizer()</code> , simplex is implemented natively in <b>DDD</b> , while subplex is ultimately called from <code>subplex::subplex()</code> .
num_cycles	Number of cycles of the optimization. When set to Inf, the optimization will be repeated until the result is, within the tolerance, equal to the starting values, with a maximum of 10 cycles.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.

take_into_account_root_edge	if TRUE, the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)
verbose	sets verbose output; default is TRUE when optimmethod is "simplex". If optimmethod is set to "simplex", then even if set to FALSE, optimizer output will be shown.
num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoe".
use_normalization	normalize the density vector during integration, more accurate but slower (default = TRUE)

## Value

Parameter estimated and maximum likelihood

## Examples

```
# Example of how to set the arguments for a ML search.
library(secsse)
library(DDD)
set.seed(13)
# 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]][c(1,4,7)] <- 4
idparslist[[2]][c(2,5,8)] <- 5
idparslist[[2]][c(3,6,9)] <- 6
masterBlock <- matrix(5,ncol = 3,nrow = 3,byrow = TRUE)
diag(masterBlock) <- NA
diff.conceal <- FALSE
idparslist[[3]] <- q_doubletrans(traits,masterBlock,diff.conceal)
startingpoint <- DDD::bd_ML(brts = ape::branching.times(phylotree))
intGuessLamba <- startingpoint$lambda0
intGuessMu <- startingpoint$mu0
idparslist <- c(1,2,3,5)
initparslist <- c(rep(intGuessLamba,3),rep((intGuessLamba/5),1))
idparsfix <- c(0,4)
parsfix <- c(0,0)
tol <- c(1e-02, 1e-03, 1e-03)
maxiter <- 1000 * round((1.25)^length(idparslist))
```

```

optimmethod <- 'simplex'
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,
  optimmethod,
  num_cycles = 1,
  verbose = FALSE)
# model$ML
# [1] -16.47099

```

---

secsse\_ml\_func\_def\_pars

*Maximum likelihood estimation for (SecSSE) complex functions as parameter*

---

## Description

Maximum likelihood estimation under Several examined and concealed States-dependent Speciation and Extinction (SecSSE) where some parameters 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 = NULL,

```

```

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)),
optimmethod = "simplex",
num_cycles = 1,
loglik_penalty = 0,
is_complete_tree = FALSE,
take_into_account_root_edge = FALSE,
num_threads = 1,
atol = 1e-08,
rtol = 1e-06,
method = "odeint::bulirsch_stoer"
)

```

## Arguments

phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
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	a numeric vector with the ID of parameters to be estimated.
initparsopt	a numeric vector with the 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	a numeric vector with the ID of the fixed parameters.
parsfix	a numeric vector with the 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. <code>id_3 &lt;- (id_1 + id_2) / 2</code> . See example.
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 for the root state: the vector <code>c(1, 0, 0)</code> indicates state 1 was the root state. When using a <code>multiPhylo</code> object, <code>root_state_weight</code> should be list where each entry in the list corresponds to the <code>root_state_weight</code> for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a <code>multiPhylo</code> object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
tol	A numeric vector with the maximum tolerance of the optimization algorithm. Default is <code>c(1e-04, 1e-05, 1e-05)</code> .
maxiter	max number of iterations. Default is <code>1000 * round((1.25) ^ length(idparsopt))</code> .
optimmethod	A string with method used for optimization. Default is "simplex". Alternative is "subplex". Both are called from <code>DDD::optimizer()</code> , simplex is implemented natively in <b>DDD</b> , while subplex is ultimately called from <code>subplex::subplex()</code> .
num_cycles	Number of cycles of the optimization. When set to <code>Inf</code> , the optimization will be repeated until the result is, within the tolerance, equal to the starting values, with a maximum of 10 cycles.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to <code>TRUE</code> , it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to <code>FALSE</code> .
take_into_account_root_edge	if <code>TRUE</code> , the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)
num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoer".

## Value

Parameter estimates and maximum likelihood

## Examples

```
# Example of how to set the arguments for an ML search. The ML search is stopped
# after 10 iterations to keep run time short.
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(5,6,5,6,5,6,5,6,5),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
# exampl3, 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-03, 1e-03, 1e-03)
optimmethod = "simplex"
cond<-"proper_cond"
root_state_weight <- "proper_weights"
sampling_fraction <- c(1,1,1)
maxiter <- 10
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,
optimmethod,
num_cycles = 1)
print(model$ML)
# ML -136.45265 if run till completion

```

---

secsse\_sim

---

*Function to simulate a tree, conditional on observing all states.*


---

## Description

By default, secsse\_sim assumes CLA-secsse simulation, e.g. inheritance of traits at speciation need not be symmetrical, and can be specified through usage of lambda-matrices. Hence, the input for lambdas is typically a list of matrices.

Simulation is performed with a randomly sampled initial trait at the crown - if you, however - want a specific, single, trait used at the crown, you can reduce the possible traits by modifying pool\_init\_states.

By default, the algorithm keeps simulating until it generates a tree where both crown lineages survive to the present - this is to ensure that the tree has a crown age that matches the used crown age. You can modify 'non-extinction' to deviate from this behaviour.

## Usage

```

secsse_sim(
  lambdas,
  mus,
  qs,
  crown_age,
  num_concealed_states,
  pool_init_states = NULL,
  sampling_fraction = NULL,
  max_spec = 1e+05,
  min_spec = 2,

```

```

max_species_extant = TRUE,
tree_size_hist = FALSE,
conditioning = "none",
non_extinction = TRUE,
verbose = FALSE,
max_tries = 1e+06,
drop_extinct = TRUE,
start_at_crown = TRUE,
seed = NULL
)

```

## Arguments

<code>lambdas</code>	speciation rates, in the form of a list of matrices.
<code>mus</code>	extinction rates, in the form of a vector.
<code>qs</code>	The Q matrix, for example the result of function <code>q_doubletrans</code> , but generally in the form of a matrix.
<code>crown_age</code>	crown age of the tree, tree will be simulated conditional on non-extinction and this crown age.
<code>num_concealed_states</code>	number of concealed states, generally equivalent to the number of examined states in the dataset.
<code>pool_init_states</code>	pool of initial states at the crown, in case this is different from all available states, otherwise leave at NULL
<code>sampling_fraction</code>	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a <code>multiPhylo</code> object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
<code>max_spec</code>	Maximum number of species in the tree (please note that the tree is not conditioned on this number, but that this is a safeguard against generating extremely large trees).
<code>min_spec</code>	Minimum number of species in the tree.
<code>max_species_extant</code>	Should the maximum number of species be counted in the reconstructed tree (if TRUE) or in the complete tree (if FALSE).
<code>tree_size_hist</code>	if TRUE, returns a vector of all found tree sizes.
<code>conditioning</code>	can be "obs_states", "true_states" or "none", the tree is simulated until one is generated that contains all observed states ("obs_states"), all true states (e.g. all combinations of obs and hidden states), or is always returned ("none"). Alternatively, a vector with the names of required observed states can be provided, e.g. <code>c("S", "N")</code> .
<code>non_extinction</code>	boolean stating if the tree should be conditioned on non-extinction of the crown lineages. Defaults to TRUE.



verbose	sets verbose output; default is TRUE when optimmethod is "simplex". If optimmethod is set to "simplex", then even if set to FALSE, optimizer output will be shown.
max_tries	maximum number of simulations to try to obtain a tree.
drop_extinct	boolean stating if extinct species should be dropped from the tree. Defaults to TRUE.
start_at_crown	if FALSE, the simulation starts with one species instead of the two assumed by default by secsse (also in ML), and the resulting crown age will be lower than the set crown age. This allows for direct comparison with BiSSE and facilitates implementing speciation effects at the crown.
seed	pseudo-random number generator seed.

**Value**

a list with four properties: phy: reconstructed phylogeny, true\_traits: the true traits in order of tip label, obs\_traits: observed traits, ignoring hidden traits and lastly: initialState, delineating the initial state at the root used.

---

secsse\_single\_branch\_loglik

*Likelihood for SecSSE model Loglikelihood calculation for the SecSSE model given a set of parameters and data, calculated for a single branch*

---

**Description**

Likelihood for SecSSE model Loglikelihood calculation for the SecSSE model given a set of parameters and data, calculated for a single branch

**Usage**

```
secsse_single_branch_loglik(
  parameter,
  phy,
  traits,
  num_concealed_states,
  cond = "proper_cond",
  root_state_weight = "proper_weights",
  sampling_fraction,
  setting_calculation = NULL,
  see_ancestral_states = FALSE,
  loglik_penalty = 0,
  is_complete_tree = FALSE,
  take_into_account_root_edge = FALSE,
  num_threads = 1,
  atol = 1e-08,
  rtol = 1e-07,
```

```

    method = "odeint::bulirsch_stoer",
    display_warning = TRUE,
    use_normalization = TRUE
  )

```

## Arguments

parameter	list where first vector represents lambdas, the second mus and the third transition rates.
phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.
traits	vector with trait states for each tip in the phylogeny. The order of the states must be the same as the tree tips. For help, see vignette("starting_secsse", package = "secsse"). When providing a multiPhylo set of multiple phylogenies, traits should be a list where each entry in the list corresponds to the matching phylogeny on that position.
num_concealed_states	number of concealed states, generally equivalent to the number of examined states in the dataset.
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 for the root state: the vector c(1, 0, 0) indicates state 1 was the root state. When using a multiPhylo object, root_state_weight should be list where each entry in the list corresponds to the root_state_weight for each tree.
sampling_fraction	vector that states the sampling proportion per trait state. It must have as many elements as there are trait states. When using a multiPhylo object, sampling fraction should be list where each entry in the list corresponds to the sampling proportion for each tree.
setting_calculation	argument used internally to speed up calculation. It should be left blank (default : setting_calculation = NULL).
see_ancestral_states	Boolean for whether the ancestral states should be shown? Defaults to FALSE.
loglik_penalty	the size of the penalty for all parameters; default is 0 (no penalty).
is_complete_tree	logical specifying whether or not a tree with all its extinct species is provided. If set to TRUE, it also assumes that all <i>all</i> extinct lineages are present on the tree. Defaults to FALSE.
take_into_account_root_edge	if TRUE, the LL integration is continued along the root edge. This also affects conditioning (as now, conditioning no longer needs to assume a speciation event at the start of the tree)

num_threads	number of threads to be used. Default is one thread.
atol	A numeric specifying the absolute tolerance of integration.
rtol	A numeric specifying the relative tolerance of integration.
method	integration method used, available are: "odeint::runge_kutta_cash_karp54", "odeint::runge_kutta_fehlberg78", "odeint::runge_kutta_dopri5", "odeint::bulirsch_stoe" and "odeint::runge_kutta4". Default method is: "odeint::bulirsch_stoer".
display_warning	display a warning if necessary
use_normalization	normalize the density vector during integration, more accurate but slower (default = TRUE)

### Value

The loglikelihood of the data given the parameter.

---

sortingtraits	<i>Data checking and trait sorting In preparation for likelihood calculation, it orders trait data according the tree tips</i>
---------------	--

---

### Description

Data checking and trait sorting In preparation for likelihood calculation, it orders trait data according the tree tips

### Usage

```
sortingtraits(trait_info, phy)
```

### Arguments

trait_info	data frame where first column has species ids and the second one is the trait associated information.
phy	phylogenetic tree of class phylo, rooted and with branch lengths. Alternatively, multiple phylogenetic trees can be provided as the multiPhylo class.

### Value

Vector of traits

### Examples

```
# Some data we have prepared
data(traits)
data('phylo_vignette')
traits <- sortingtraits(traits, phylo_vignette)
```

---

traits	<i>A table with trait info to run the vignette</i>
--------	--

---

**Description**

An example of trait information in the right format for secsse

**Usage**

traits

**Format**

A data frame where each species has a trait state associated

# Index

\* **datasets**  
  example\_phy\_GeoSSE, [18](#)  
  phylo\_vignette, [21](#)  
  traits, [44](#)

cla\_id\_paramPos, [3](#)  
cla\_secsse\_loglik, [3](#)  
cla\_secsse\_ml, [6](#)  
cla\_secsse\_ml(), [20](#)  
cla\_secsse\_ml\_func\_def\_pars, [9](#)  
create\_default\_lambda\_transition\_matrix,  
  [13](#)  
create\_default\_shift\_matrix, [14](#)  
create\_lambda\_list, [15](#)  
create\_mu\_vector, [16](#)  
create\_q\_matrix, [17](#)  
create\_q\_matrix(), [14](#)

DDD::optimizer(), [8](#), [11](#), [33](#), [37](#)

event\_times, [18](#)  
example\_phy\_GeoSSE, [18](#)  
expand\_q\_matrix, [19](#)  
expand\_q\_matrix(), [26](#)  
extract\_par\_vals, [19](#)

fill\_in, [20](#)

id\_paramPos, [21](#)

phylo\_vignette, [21](#)  
plot\_idparslist, [22](#)  
plot\_state\_exact, [23](#)  
prepare\_full\_lambdas, [25](#)

q\_doubletrans, [26](#)  
q\_doubletrans(), [19](#)

secsse\_loglik, [27](#)  
secsse\_loglik\_eval, [30](#)  
secsse\_ml, [32](#)  
secsse\_ml(), [21](#)  
secsse\_ml\_func\_def\_pars, [35](#)  
secsse\_sim, [39](#)  
secsse\_single\_branch\_loglik, [41](#)  
sortingtraits, [43](#)  
subplex::subplex(), [8](#), [11](#), [33](#), [37](#)  
traits, [44](#)