

# Package ‘CRABS’

October 12, 2022

**Title** Congruent Rate Analyses in Birth-Death Scenarios

**Version** 1.1.0

**Encoding** UTF-8

**Description** Features tools for exploring congruent phylogenetic birth-death models. It can construct the pulled speciation- and net-diversification rates from a reference model. Given alternative speciation- or extinction rates, it can construct new models that are congruent with the reference model. Functionality is included to sample new rate functions, and to visualize the distribution of one congruence class. See also Louca & Pennell (2020) <[doi:10.1038/s41586-020-2176-1](https://doi.org/10.1038/s41586-020-2176-1)>.

**LazyData** true

**Depends** R (>= 3.5.0), ggplot2

**Imports** magrittr, deSolve, dplyr, tibble, colorspace, patchwork,  
latex2exp, tidyr, pracma, ape

**License** GPL-3

**Suggests** knitr, rmarkdown

**RoxygenNote** 7.2.0

**URL** <https://github.com/afmagee/CRABS>

**NeedsCompilation** no

**Author** Bjørn Tore Kopperud [aut, cre],  
Sebastian Höhna [aut],  
Andrew F. Magee [aut]

**Maintainer** Bjørn Tore Kopperud <kopperud@protonmail.com>

**Repository** CRAN

**Date/Publication** 2022-06-20 16:30:02 UTC

## R topics documented:

CRABS-package	2
congruent.models	3
crabs.loglikelihood	4

create.model . . . . .	5
model2df . . . . .	6
plot.CRABS . . . . .	6
plot.CRABSset . . . . .	7
primates . . . . .	8
primates_ebd . . . . .	8
primates_ebd_log . . . . .	9
primates_ebd_tess . . . . .	9
primates_ebd_treepar . . . . .	10
print.CRABS . . . . .	10
print.CRABSposterior . . . . .	11
print.CRABSset . . . . .	11
print.CRABSsets . . . . .	12
read.RevBayes . . . . .	13
sample.basic.models . . . . .	14
sample.congruence.class . . . . .	15
sample.congruence.class.posterior . . . . .	16
sample.rates . . . . .	18
summarize.posterior . . . . .	19
summarize.trends . . . . .	20

<b>Index</b>	<b>22</b>
--------------	-----------

---

CRABS-package

*CRABS: Congruent Rate Analyses in Birth-death Scenarios*

---

## Description

Features tools for exploring congruent phylogenetic birth-death models. It can construct the pulled speciation- and net-diversification rates from a reference model. Given alternative speciation- or extinction rates, it can construct new models that are congruent with the reference model. Functionality is included to sample new rate functions, and to visualize the distribution of one congruence class. See also Louca & Pennell (2020) doi: [10.1038/s4158602021761](https://doi.org/10.1038/s4158602021761).

## References

- Louca, S., & Pennell, M. W. (2020). Extant timetrees are consistent with a myriad of diversification histories. *Nature*, 580(7804), 502-505.

## Author(s)

**Maintainer:** Bjørn Tore Kopperud <kopperud@protonmail.com>

Authors:

- Sebastian Höhna
- Andrew F. Magee

**See Also**

Useful links:

- <https://github.com/afmagee/CRABS>

---

congruent.models      *Create a set of congruent models*

---

**Description**

Create a set of congruent models

**Usage**

```
congruent.models(
  model,
  mus = NULL,
  lambdas = NULL,
  keep_ref = TRUE,
  ode_solver = TRUE
)
```

**Arguments**

model	The reference model. An object of class "CRABS"
mus	A list of extinction-rate functions
lambdas	A list of speciation-rate functions
keep_ref	Whether or not to keep the reference model in the congruent set
ode_solver	Whether to use a numerical ODE solver to solve for lambda

**Value**

An object of class "CRABSset"

**Examples**

```
data(primates_ebd)
lambda <- approxfun(primates_ebd$time, primates_ebd$lambda)
mu <- approxfun(primates_ebd$time, primates_ebd$mu)

## A reference model
times <- seq(0, max(primates_ebd$time), length.out = 500)
model <- create.model(lambda, mu, times = times)

mu1 <- lapply(c(0.5, 1.5, 3.0), function(m) function(t) m)
```

```
model_set1 <- congruent.models(model, mus = mu1)

model_set1

lambda0 <- lambda(0.0) ## Speciation rates must all be equal at the present
bs <- c(0.0, 0.01, 0.02)
lambda1 <- lapply(bs, function(b) function(t) lambda0 + b*t)

model_set2 <- congruent.models(model, lambdas = lambda1)

model_set2
```

---

crabs.loglikelihood    *Compute likelihood*

---

## Description

Compute likelihood

## Usage

```
crabs.loglikelihood(phy, model, rho = 1)
```

## Arguments

phy	an object of class "phylo"
model	an object of class "CRABS"
rho	the taxon sampling fraction

## Value

the log-likelihood of the tree given the model

## Examples

```
library(ape)
lambda <- function(t) exp(0.3*t) - 0.5*t
mu <- function(t) exp(0.3*t) - 0.2*t - 0.8

model <- create.model(lambda, mu, times = seq(0, 3, by = 0.005))

set.seed(123)
phy <- rcoal(25)

crabs.loglikelihood(phy, model)
```

---

create.model	<i>Computes the congruent class, i.e., the pulled rates.</i>
--------------	--

---

### Description

Computes the congruent class, i.e., the pulled rates.

### Usage

```
create.model(
  func_spec0,
  func_ext0,
  times = seq(from = 0, to = 5, by = 0.005),
  func_p_spec = NULL,
  func_p_div = NULL
)
```

### Arguments

func_spec0	The speciation rate function (measured in time before present).
func_ext0	The extinction rate function (measured in time before present).
times	the time knots for the piecewise-linear rate functions
func_p_spec	the pulled speciation rate function
func_p_div	the pulled net-diversification rate function

### Value

A list of rate functions representing this congruence class.

### Examples

```
lambda1 <- function(t) exp(0.3*t) - 0.5*t + 1
mu1 <- function(t) exp(0.3*t) - 0.2*t + 0.2

model1 <- create.model(lambda1, mu1, times = seq(0, 5, by = 0.005))

model1

data("primates_ebd")

lambda2 <- approxfun(primates_ebd[["time"]], primates_ebd[["lambda"]])
mu2 <- approxfun(primates_ebd[["time"]], primates_ebd[["mu"]])
model2 <- create.model(lambda2, mu2, primates_ebd[["time"]])

model2
```

---

model2df	<i>model2df</i>
----------	-----------------

---

**Description**

model2df

**Usage**

```
model2df(model, gather = TRUE, rho = 1, compute.pulled.rates = TRUE)
```

**Arguments**

model	an object of class "CRABS"
gather	boolean. Whether to return wide or long data frame
rho	the sampling fraction at the present. Used to calculate the pulled speciation rate
compute.pulled.rates	whether to compute the pulled rates

**Value**

a data frame

**Examples**

```
lambda <- function(t) 2.0 + sin(0.8*t)
mu <- function(t) 1.5 + exp(0.15*t)
times <- seq(from = 0, to = 4, length.out = 1000)
model <- create.model(lambda, mu, times = times)

model2df(model)
```

---

plot.CRABS	<i>Plots the rate functions including the pulled rates.</i>
------------	---

---

**Description**

Plots the rate functions including the pulled rates.

**Usage**

```
## S3 method for class 'CRABS'
plot(x, ...)
```

### Arguments

x                    An object of class "CRABS"  
...                   other parameters

### Value

a patchwork object

### Examples

```
data(primates_ebd)
lambda <- approxfun(primates_ebd$time, primates_ebd$lambda)
mu <- approxfun(primates_ebd$time, primates_ebd$mu)
times <- seq(0, max(primates_ebd$time), length.out = 500)

model <- create.model(lambda, mu, times = times)

plot(model)
```

---

plot.CRABSset                    *Plots the rate functions*

---

### Description

Plots the rate functions

### Usage

```
## S3 method for class 'CRABSset'
plot(x, ...)
```

### Arguments

x                    A list of congruent birth-death x  
...                   other parameters

### Value

a patchwork object object

**Examples**

```

data(primates_ebd)
lambda <- approxfun(primates_ebd$time, primates_ebd$lambda)
mu <- approxfun(primates_ebd$time, primates_ebd$mu)
times <- seq(0, max(primates_ebd$time), length.out = 500)

model <- create.model(lambda, mu, times = times)

mus <- list(function(t) 0.2 + exp(0.01*t),
            function(t) 0.2 + sin(0.35*t) + 0.1*t,
            function(t) 1.0,
            function(t) 0.5 + 0.2*t)
models <- congruent.models(model, mus = mus)

plot(models)

```

---

primates

*Primates phylogenetic tree*


---

**Description**

The example tree taken from the RevBayes tutorial website

**Usage**

```
data(primates)
```

**Format**

An object of class `phylo` of length 5.

---

primates\_ebd

*RevBayes Primates birth-death model*


---

**Description**

The results of a bayesian horseshoe markov random field (HSMRF) episodic birth-death model, fitted on the primates tree. One hundred episodes. Each estimate is the posterior median. The time unit is millions of years before the present.

**Usage**

```
data(primates_ebd)
```

**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 100 rows and 3 columns.



---

primates_ebd_log	<i>Primates birth-death model</i>
------------------	-----------------------------------

---

**Description**

See ?primates\_ebd, but including posterior samples instead of a summary.

**Usage**

```
data(primates_ebd_log)
```

**Format**

An object of class tbl\_df (inherits from tbl, data.frame) with 251 rows and 604 columns.

---

primates_ebd_tess	<i>TESS Primates birth-death model</i>
-------------------	--

---

**Description**

The results of a bayesian episodic birth-death model in the R-package TESS, fitted on the primates tree. One hundred episodes. Each estimate is the posterior median. The time unit is millions of years before the present.

**Usage**

```
data(primates_ebd_tess)
```

**Format**

An object of class tbl\_df (inherits from tbl, data.frame) with 100 rows and 3 columns.

---

```
primates_ebd_treepar  TreePar Primates birth-death model
```

---

**Description**

The results of a birth-death model in the R-package TreePar, fitted on the primates tree. The estimated model has two epochs, that are maximum-likelihood estimates. The time unit is millions of years before the present.

**Usage**

```
data(primates_ebd_treepar)
```

**Format**

An object of class tbl\_df (inherits from tbl, data.frame) with 100 rows and 3 columns.

---

```
print.CRABS          Print method for CRABS object
```

---

**Description**

Print method for CRABS object

**Usage**

```
## S3 method for class 'CRABS'
print(x, ...)
```

**Arguments**

```
x          and object of class CRABS
...        other arguments
```

**Value**

nothing

**Examples**

```
data(primates_ebd)
lambda <- approxfun(primates_ebd$time, primates_ebd$lambda)
mu <- approxfun(primates_ebd$time, primates_ebd$mu)
times <- seq(0, max(primates_ebd$time), length.out = 500)

model <- create.model(lambda, mu, times = times)

print(model)
```

---

print.CRABSPosterior    *Title*

---

**Description**

Title

**Usage**

```
## S3 method for class 'CRABSPosterior'  
print(x, ...)
```

**Arguments**

x                    a list of CRABS objects  
...                   additional parameters

**Value**

nothing

**Examples**

```
data(primates_ebd_log)  
posterior <- read.RevBayes(primates_ebd_log, max_t = 65, n_samples = 20)  
print(posterior)
```

---

print.CRABSSet            *Print method for CRABSSet object*

---

**Description**

Print method for CRABSSet object

**Usage**

```
## S3 method for class 'CRABSSet'  
print(x, ...)
```

**Arguments**

x                    an object of class CRABSSet  
...                   other arguments



```

rate.type = "extinction",
rate0.median = 0.1,
model = "MRF",
max.rate = 1.0)

print(samples)

```

---

```
read.RevBayes      read RevBayes log file
```

---

## Description

read RevBayes log file

## Usage

```
read.RevBayes(x, n_times, max_t = 100, n_samples = 20, summary_type = "none",
extinction_prefix = "extinction_rate.", speciation_prefix = "speciation_rate.")
```

## Arguments

x	path to log, or data frame
n_times	number of time knots
max_t	tree height
n_samples	first n posterior samples
summary_type	either "none" for all the posterior samples, or "mean" or "median" for the posterior mean/median
extinction_prefix	the prefix string for the extinction rate column names. Must be unique
speciation_prefix	the prefix string for the speciation rate column names. Must be unique

## Value

a set of CRABS models, each being a sample in the posterior

## Examples

```
data(primates_ebd_log)
posterior <- read.RevBayes(primates_ebd_log, n_times = 500, max_t = 65, n_samples = 20)
```

---

sample.basic.models    *Samples simple increase/decrease models through time with noise.*

---

### Description

Samples simple increase/decrease models through time with noise.

### Usage

```
sample.basic.models(
  times,
  rate0 = NULL,
  model = "exponential",
  direction = "decrease",
  noisy = TRUE,
  MRF.type = "HSMRF",
  monotonic = FALSE,
  fc.mean = 3,
  rate0.median = 0.1,
  rate0.logsd = 1.17481,
  mrf.sd.scale = 1,
  min.rate = 0,
  max.rate = 10
)
```

### Arguments

times	the time knots
rate0	The rate at present, otherwise drawn randomly.
model	"MRF" for pure MRF model, otherwise MRF has a trend of type "exponential", "linear", or "episodic<n>"
direction	"increase" or "decrease" (measured in past to present)
noisy	If FALSE, no MRF noise is added to the trajectory
MRF.type	"HSMRF" or "GMRF", type for stochastic noise.
monotonic	Whether the curve should be forced to always move in one direction.
fc.mean	Determines the average amount of change when drawing from the model.
rate0.median	When not specified, rate at present is drawn from a lognormal distribution with this median.
rate0.logsd	When not specified, rate at present is drawn from a lognormal distribution with this sd
mrf.sd.scale	scale the sd of the mrf process up or down. defaults to 1.0
min.rate	The minimum rate (rescaling done after drawing rates).
max.rate	The maximum rate (rescaling done after drawing rates).

**Value**

Speciation or extinction rate at a number of timepoints.

**Examples**

```
data("primates_ebd")

l <- approxfun(primates_ebd[["time"]], primates_ebd[["lambda"]])
mu <- approxfun(primates_ebd[["time"]], primates_ebd[["mu"]])
times <- primates_ebd[["time"]]

model <- create.model(l, mu, times)

mus <- sample.basic.models(times = times,
                           rate0 = 0.05,
                           "MRF",
                           MRF.type = "HSMRF",
                           fc.mean = 2.0,
                           min.rate = 0.0,
                           max.rate = 1.0)

model_set <- congruent.models(model, mus = mus)

model_set
```

---

sample.congruence.class

*Stochastic exploration of congruent models.*

---

**Description**

Stochastic exploration of congruent models.

**Usage**

```
sample.congruence.class(
  model,
  num.samples,
  rate.type = "both",
  sample.speciation.rates = NULL,
  sample.extinction.rates = NULL
)
```

**Arguments**

model	the reference model, an object of class "CRABS"
num.samples	The pulled diversification rate function (measured in time before present).
rate.type	either "extinction", "speciation", or "both"

sample.speciation.rates  
     a function that when called returns a speciation rate function

sample.extinction.rates  
     a function that when called returns a extinction rate function

### Value

A named list with congruent rates.

### Examples

```
data("primates_ebd")

l <- approxfun(primates_ebd[["time"]], primates_ebd[["lambda"]])
mu <- approxfun(primates_ebd[["time"]], primates_ebd[["mu"]])
times <- primates_ebd[["time"]]

model <- create.model(l, mu, primates_ebd[["time"]])

extinction_rate_samples <- function(){
  res <- sample.basic.models(times = times,
                             rate0 = 0.05,
                             model = "MRF",
                             MRF.type = "HSMRF",
                             fc.mean = 2.0,
                             min.rate = 0.0,
                             max.rate = 1.0)

  return(res)
}

samples <- sample.congruence.class(model,
                                   num.samples = 8,
                                   rate.type = "extinction",
                                   sample.extinction.rates = extinction_rate_samples)
```

---

sample.congruence.class.posterior

*Stochastic exploration of congruent models for all samples in the posterior*

---

### Description

This function takes a posterior sample as input: a list of CRABS objects. It will then iterate over the samples, and for each posterior sample it will sample from the posterior class. It will sample using the `sample.basic.models` function, and all additional parameters are passed to `sample.basic.models`.



**Usage**

```
sample.congruence.class.posterior(
  posterior,
  num.samples,
  rate.type = "extinction",
  mu0.equal = FALSE,
  rate0 = NULL,
  ...
)
```

**Arguments**

posterior	a list of CRABS model objects
num.samples	The pulled diversification rate function (measured in time before present).
rate.type	either "extinction", "speciation", or "both"
mu0.equal	whether to propose alternative mu starting at mu0 equal to the posterior sample. default to FALSE
rate0	rate0 allows the user to fix the extinction rate at the present to a single value. defaults to NULL, for drawing it randomly
...	Arguments passed on to <a href="#">sample.basic.models</a>
	times the time knots
	model "MRF" for pure MRF model, otherwise MRF has a trend of type "exponential", "linear", or "episodic<n>"
	direction "increase" or "decrease" (measured in past to present)
	noisy If FALSE, no MRF noise is added to the trajectory
	MRF.type "HSMRF" or "GMRF", type for stochastic noise.
	monotonic Whether the curve should be forced to always move in one direction.
	fc.mean Determines the average amount of change when drawing from the model.
	rate0.median When not specified, rate at present is drawn from a lognormal distribution with this median.
	rate0.logsd When not specified, rate at present is drawn from a lognormal distribution with this sd
	mrf.sd.scale scale the sd of the mrf process up or down. defaults to 1.0
	min.rate The minimum rate (rescaling done after drawing rates).
	max.rate The maximum rate (rescaling done after drawing rates).

**Value**

A named list with congruent rates.

**Examples**

```

data(primates_ebd_log)

posterior <- read.RevBayes(primates_ebd_log, max_t = 65, n_samples = 10)

samples <- sample.congruence.class.posterior(posterior,
                                             num.samples = 5,
                                             rate.type = "extinction",
                                             rate0.median = 0.1,
                                             model = "MRF",
                                             max.rate = 1.0)

print(samples)

```

---

sample.rates

*Sample custom functions through time.*


---

**Description**

Sample custom functions through time.

**Usage**

```

sample.rates(
  times,
  lambda0 = NULL,
  rsample = NULL,
  rsample0 = NULL,
  autocorrelated = FALSE
)

```

**Arguments**

times	the time knots
lambda0	The rate at present
rsample	Function to sample next rate
rsample0	Function to sample rate at present
autocorrelated	Should rates be autocorrelated?

**Value**

Sampled rate vector

**Examples**

```

data("primates_ebd")

l <- approxfun(primates_ebd[["time"]], primates_ebd[["lambda"]])
mu <- approxfun(primates_ebd[["time"]], primates_ebd[["mu"]])
times <- primates_ebd[["time"]]

model <- create.model(l, mu, times)

rsample <- function(n) runif(n, min = 0.0, max = 0.9)
mu <- sample.rates(times, 0.5, rsample = rsample)

model_set <- congruent.models(model, mus = mu)

model_set

```

---

summarize.posterior     *Summarize trends in the posterior*

---

**Description**

Summarize trends in the posterior

**Usage**

```

summarize.posterior(posterior, threshold = 0.01, rate_name = "lambda",
return_data = FALSE, rm_singleton = FALSE, per_time = TRUE,
window_size = 1, relative_deltas = FALSE)

```

**Arguments**

posterior	a list of CRABS objects, each one representing a sample from the posterior
threshold	a threshold for when $\Delta\lambda_i$ should be interpreted as decreasing, flat, or increasing
rate_name	either "lambda" or "mu" or "delta"
return_data	instead of plots, return the plotting dataframes
rm_singleton	whether or not to remove singletons. Pass starting at present, going towards ancient
per_time	whether to compute $\Delta\lambda_i$ that are in units of per time, i.e. divide by $\Delta t$
window_size	the window size "k" in $\Delta\lambda_i = \lambda_i - \lambda(i - k)$
relative_deltas	whether to divide $\Delta\lambda_i$ by the local lambda value

**Value**

a ggplot object

**Examples**

```

data(primates_ebd_log)

posterior <- read.RevBayes(primates_ebd_log, max_t = 65, n_samples = 10)

samples <- sample.congruence.class.posterior(posterior,
                                             num.samples = 5,
                                             rate.type = "extinction",
                                             rate0.median = 0.1,
                                             model = "MRF",
                                             max.rate = 1.0)

p <- summarize.posterior(samples, threshold = 0.05)

```

---

<code>summarize.trends</code>	<i>Summarize trends in the congruence class</i>
-------------------------------	---

---

**Description**

Summarize trends in the congruence class

**Usage**

```

summarize.trends(model_set, threshold = 0.005, rate_name = "lambda",
                 window_size = 1, method = "neighbour", per_time = TRUE, return_data = FALSE,
                 rm_singleton = FALSE, relative_deltas = FALSE, group_names = NULL)

```

**Arguments**

<code>model_set</code>	an object of type "CRABSset"
<code>threshold</code>	a threshold for when $\Delta\lambda_i$ should be interpreted as decreasing, flat, or increasing
<code>rate_name</code>	either "lambda" or "mu" or "delta"
<code>window_size</code>	the window size "k" in $\Delta\lambda_i = \lambda_i - \lambda(i - k)$
<code>method</code>	default to "neighbour", i.e. to compare rate values at neighbouring time points.
<code>per_time</code>	whether to compute $\Delta\lambda_i$ that are in units of per time, i.e. divide by $\Delta t$
<code>return_data</code>	instead of plots, return the plotting dataframes
<code>rm_singleton</code>	whether or not to remove singletons. Pass starting at present, going towards ancient
<code>relative_deltas</code>	whether to divide $\Delta\lambda_i$ by the local lambda value
<code>group_names</code>	a vector of prefixes, if you want to group the models in a facet. For example 'c("reference", "model")'

**Value**

a patchwork object

**Examples**

```
data(primates_ebd)
lambda <- approxfun(primates_ebd$time, primates_ebd$lambda)
mu <- approxfun(primates_ebd$time, primates_ebd$mu)
times <- seq(0, max(primates_ebd$time), length.out = 500)

reference <- create.model(lambda, mu, times = times)

mus <- list(function(t) exp(0.01*t) - 0.01*t - 0.9,
            function(t) exp(-0.02*t) - 0.2,
            function(t) exp(-0.07*t) + 0.02*t - 0.5,
            function(t) 0.2 + 0.01*t,
            function(t) 0.2)

model_set <- congruent.models(reference, mus = mus)

p <- summarize.trends(model_set, 0.02)
```

# Index

## \* datasets

- primates, [8](#)
- primates\_ebd, [8](#)
- primates\_ebd\_log, [9](#)
- primates\_ebd\_tess, [9](#)
- primates\_ebd\_treepar, [10](#)

[congruent.models](#), [3](#)

[CRABS \(CRABS-package\)](#), [2](#)

[CRABS-package](#), [2](#)

[crabs.loglikelihood](#), [4](#)

[create.model](#), [5](#)

[model2df](#), [6](#)

[plot.CRABS](#), [6](#)

[plot.CRABSset](#), [7](#)

primates, [8](#)

primates\_ebd, [8](#)

primates\_ebd\_log, [9](#)

primates\_ebd\_tess, [9](#)

primates\_ebd\_treepar, [10](#)

[print.CRABS](#), [10](#)

[print.CRABSposterior](#), [11](#)

[print.CRABSset](#), [11](#)

[print.CRABSsets](#), [12](#)

[read.RevBayes](#), [13](#)

[sample.basic.models](#), [14](#), [16](#), [17](#)

[sample.congruence.class](#), [15](#)

[sample.congruence.class.posterior](#), [16](#)

[sample.rates](#), [18](#)

[summarize.posterior](#), [19](#)

[summarize.trends](#), [20](#)