# Package ‘CRABS’

October 24, 2023

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

**Version** 1.2.0

**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>.

**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** 7.2.3

**URL** `https://github.com/afmagee/CRABS`

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2023-10-24 10:20:07 UTC

## R topics documented:

`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.

## References

- Louca, S., & Pennell, M. W. (2020). Extant timetrees are consistent with a myriad of diversification histories. Nature, 580(7804), 502-505. https://doi.org/10.1038/s41586-020-2176-1
- Höhna, S., Kopperud, B. T., & Magee, A. F. (2022). CRABS: Congruent rate analyses in birth–death scenarios. Methods in Ecology and Evolution, 13, 2709– 2718. https://doi.org/10.1111/2041-210X.13997
- Kopperud, B. T., Magee, A. F., & Höhna, S. (2023). Rapidly Changing Speciation and Extinction Rates Can Be Inferred in Spite of Nonidentifiability. Proceedings of the National Academy of Sciences 120 (7): e2208851120. https://doi.org/10.1073/pnas.2208851120

- Andréoletti, J. & Morlon, H. (2023). Exploring congruent diversification histories with flexibility and parsimony. Methods in Ecology and Evolution. https://doi.org/10.1111/2041-210X.14240

## Author(s)

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

Authors:

- Sebastian Höhna
- Andrew F. Magee
- Jérémy Andréoletti

## 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` *Computes the congruent class, i.e., the pulled rates.*

---

### 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
```

---

`full.plot.regularity.thresholds`
*Plots the rate functions after filtering them according to a given penalty and predefined thresholds.*

---

## Description

Plots the rate functions after filtering them according to a given penalty and predefined thresholds.

## Usage

```
full.plot.regularity.thresholds(
  samples,
  filtering_fractions = c(0.01, 0.05, 0.2, 0.9),
  penalty = "L1",
  rates = c("lambda", "mu")
)
```

## Arguments

| | |
|---|---|
| `samples` | A list of (congruent) CRABS models |
| `filtering_fractions` | A vector of thresholds for filtering, as fractions of the most regular trajectories. |
| `penalty` | The choice of penalty, among "L1", "L2" and "L1_derivative" (penalty on derivative shifts). |
| `rates` | A vector of rate(s) to be plotted, among "lambda" (speciation), "mu" (extinction), "delta" (net-diversification) and "epsilon" (turnover). |

## Value

Plots an array of rate trajectories for the chosen rates and thresholds.

## Examples

```
data("primates_ebd")
set.seed(123)

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)

sample.joint.rates <- function(n) {
  sample.basic.models.joint(times = times,
                            p.delta = model$p.delta,
                            beta.param = c(0.5,0.3),
                            lambda0 = l(0.0),
                            mu0.median = mu(0.0))
}

joint.samples <- sample.congruence.class(model = model,
                                         num.samples = 100,
                                         rate.type = "joint",
                                         sample.joint.rates = sample.joint.rates)

full.plot.regularity.thresholds(joint.samples)
```

---

`joint.congruent.models`
*Create a set of congruent models*

---

## Description

Create a set of congruent models

## Usage

```
joint.congruent.models(model, mus, lambdas, keep_ref = 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 |

## Value

An object of class "CRABSset"

## Examples

```
# This function should not have to be used externally.
# It is called in the CRABS function `sample.congruence.class` when `rate.type=="joint"`.
```

---

`model2df` *model2df*

---

## 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` *Plots the rate functions including the pulled rates.*

### 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` *Primates birth-death model*

---

## 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` *TESS Primates birth-death model*

### 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 |

## 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)

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)

print(models)
```

---

`print.CRABSsets` *print.CRABSsets*

---

## Description

print.CRABSsets

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a list of (congruent) CRABS sets |
| `...` | additional parameters |

## Value

nothing

## 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)
```

## `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 fone after after drawing rates). |
| `max.rate` | The maximum rate (rescaling fone after 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.basic.models.joint` *Jointly samples speciation and extinction trajectories through time, with noise.*

---

## Description

Jointly samples speciation and extinction trajectories through time, with noise.

## Usage

```
sample.basic.models.joint(
  times,
  p.delta,
  lambda0,
  mu0 = NULL,
  MRF.type = "HSMRF",
  beta.param = c(0.3, 0.3),
  mu0.median = 0.1,
  mu0.logsd = 1.17481,
  mrf.sd.scale = 1,
  min.lambda = 0,
  min.mu = 0,
  max.lambda = 10,
  max.mu = 10,
  min.p = -0.05,
  max.p = 1.05
)
```

## Arguments

| | |
|---|---|
| `times` | the time knots |
| `p.delta` | The The pulled diversification rate function (measured in time before present). |
| `lambda0` | The speciation rate at present. |
| `mu0` | The extinction rate at present, otherwise drawn randomly. |
| `MRF.type` | "HSMRF" or "GMRF", type for stochastic noise. |
| `beta.param` | Parameters of the Beta distribution used for |
| `mu0.median` | When not specified, extinction rate at present is drawn from a lognormal distribution with this median. |
| `mu0.logsd` | When not specified, extinction 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.lambda` | The minimum speciation rate (rescaling done after after drawing rates). |
| `min.mu` | The minimum extinction rate (rescaling done after after drawing rates). |
| `max.lambda` | The maximum speciation rate (rescaling done after after drawing rates). |
| `max.mu` | The maximum extinction rate (rescaling done after after drawing rates). |
| `min.p` | The lower bound of parameter p's trajectory. |
| `max.p` | The upper bound of parameter p's trajectory. |

## 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)

sample.joint.rates <- function(n) {
  sample.basic.models.joint(times = times,
                            p.delta = model$p.delta,
                            beta.param = c(0.5,0.3),
                            lambda0 = l(0.0),
                            mu0.median = mu(0.0))
}

joint.samples <- sample.congruence.class(model = model,
                                         num.samples = 40,
                                         rate.type = "joint",
                                         sample.joint.rates = sample.joint.rates)

joint.samples
```

## 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,
  sample.joint.rates = NULL
)
```

### Arguments

| | |
|---|---|
| `model` | the reference model, an object of class "CRABS" |
| `num.samples` | The number of samples to be drawn |
| `rate.type` | either "extinction", "speciation", "both" or "joint" |
| `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 |
| `sample.joint.rates` | a function that when called returns a list with a speciation rate function and an 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"]])

# Sampling extinction rates
```

```
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)

samples

# Jointly sampling speciation and extinction rates

sample.joint.rates <- function(n) {
  sample.basic.models.joint(times = times,
                            p.delta = model$p.delta,
                            beta.param = c(0.5,0.3),
                            lambda0 = l(0.0),
                            mu0.median = mu(0.0))
}

joint.samples <- sample.congruence.class(model = model,
                                         num.samples = 40,
                                         rate.type = "joint",
                                         sample.joint.rates = sample.joint.rates)

joint.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 number of samples to be drawn |
| `rate.type` | either "extinction", "speciation", "both" or "joint" |
| `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 `sample.basic.models` |

- `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 fone after after drawing rates).
- `max.rate` The maximum rate (rescaling fone after 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)
```

---

`summarize.trends` *Summarize trends in the congruence class*

---

## 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

| | |
|---|---|
| `model_set` | an object of type "CRABSset" |
| `threshold` | a threshold for when $\Delta\lambda i$ should be interpreted as decreasing, flat, or increasing |
| `rate_name` | either "lambda" or "mu" or "delta" |
| `window_size` | the window size "k" in $\Delta\lambda i = \lambda i - \lambda(i-k)$ |
| `method` | default to "neighbour", i.e. to compare rate values at neighbouring time points. |
| `per_time` | whether to compute $\Delta\lambda i$ that are in units of per time, i.e. divide by $\Delta t$ |
| `return_data` | instead of plots, return the plotting dataframes |
| `rm_singleton` | whether or not to remove singletons. Pass starting at present, going towards ancient |
| `relative_deltas` | whether to divide $\Delta\lambda i$ by the local lambda value |
| `group_names` | 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