Package 'menura'

November 26, 2018

Title Fitting (Non)-Gaussian diffusion models to phylogenies

Version 0.4.1

Description Fits user-defined stochastic diffusion models to univariate trait data on a phylogeny. Also fits three canned models: Ornstein Uhlenbeck (OU; Gaussian), Cox, Ingersoll, Ross (CIR; Non-Gaussian) and an OU-like model with a Beta stationary distribution (Beta, Non-Gaussian). Models are fitted and parameters are estimated using a Data Augmentation - Metropolis Hastings algorithm. Output can be analysed using the 'coda' package , 'tidybayes' package or other appropriate packages for Bayesian analysis and visualisation. Menura is the genus name for the Australian Superb Lyrebird (Menura novaehollandiae), known for being the world's largest passerine, its elaborate tail and its talent for mimicry. We wrote menura in order to study the behaviour of SDE models on phylogenies. Thus, menura is EXPERIMENTAL SOFTWARE. If you are unsure whether menura is appropriate for your own analyses, it probably isn't.

Author Simone Blomberg <s.blomberg1@uq.edu.au>

Maintainer Simone Blomberg <s.blomberg1@uq.edu.au>

Depends R (>= 3.2.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports ape, sde, stats, graphics

NeedsCompilation no

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

fit_model *Bayesian Estimator of Parameters of Univariate Diffusion Models for Continuous Trait Evolution*

Description

This function estimates posterior distributions for evolutionary models of continuous traits on a phylogeny. The evolutionary processes considered here belong to a class of diffusion processes which are typically given as solutions to the stochastic differential equations of the form given by

$$dX_t = b(X_t, \alpha, \mu)dt + \sigma(X_t, \epsilon)dW_t, \;\; X_0 = x_0$$

where X_t denote the state variable (ie the trait value), t the time, the drift function b and the diffusion function σ are known in parametric form and where α, μ, and ϵ are the parameters, and W_t is Brownian motion. The value of X_t at time t_0, X_0, is independent of the W_t. If σ is a constant, we set $\epsilon = \sigma$.

Usage

```
fit_model(tr, tipdata, rt_value, model, priors, proposals,
  mcmc_type = "tanner-wong", alpha = NULL, mu = NULL, sigma = NULL,
  N=1000, init_method="sim", update_method="subtree", iters=5000,
  method = "euler", fossils=NULL, ...)

## Default S3 method:
fit_model(tr, tipdata, rt_value = mean(tipdata),
  model = "OU",
  priors = list(
    alpha = list (df =  function(x, a = 1, b = 125, log_scale = TRUE) {
                            dunif(x, min = a, max = b, log = log_scale)},
                  rf = function(n, a = 1, b = 125) {
                            runif(n, min = a, max = b)} ),
    mu = list (df = function(x, a = 0, b = 20, log_scale = TRUE) {
                            dnorm(x, mean = a, sd = b, log = log_scale)},
               rf = function(n, a = 0, b = 20) {
                            rnorm(n, mean = a, sd = b)}),
    sigma = list (df = function(x, a = 1, b = 225, log_scale = TRUE) {
                            dunif(x, min = a, max = b, log = log_scale) },
                   rf = function(n, a = 1, b = 225) {
                            runif(n, min = a, max = b)})
  ),
  proposals = list(
    alpha = list (df = function(n, alpha, gamma = 0.5, log_scale = TRUE) {
                    dlnorm(n, meanlog = log(alpha), sdlog = gamma,
                    log = log_scale) },
                  rf = function(n, alpha, gamma = 0.5) {
                    rlnorm(n, meanlog = log(alpha), sdlog = gamma) }),
```

```
    mu = list (df = function(n, mu, gamma = 0.5, log_scale = TRUE) {
                    dnorm(n, mean = mu, sd = gamma, log = log_scale)},
               rf = function(n, mu, gamma = 0.5) {
                    rnorm(n, mean = mu, sd = gamma)}),
    sigma = list(df = function(n, sigma, gamma = 0.5, log_scale = TRUE) {
                    dlnorm(n, meanlog = log(sigma), sdlog = gamma,
                    log = log_scale)},
                 rf = function(n, sigma, gamma = 0.5) {
                    rlnorm(n, meanlog = log(sigma), sdlog = gamma)})
  ),
  mcmc_type = "tanner-wong", alpha = NULL, mu = NULL, sigma = NULL,
  N = 1000, init_method = "sim", update_method = "subtree", iters = 5000,
  method = "euler", fossils = NULL, ...)
```

Arguments

tr	single evolutionary tree as an object of class 'phylo' in package ape.
tipdata	a numeric vector containing values of the trait at the tips. These must be in the same order as those in tr$tip.label.
rt_value	value of the trait at the root.
model	either a list containing the drift and diffusion coefficients in quote format as functions of alpha, mu and sigma, or a string ("OU", "CIR", or "Beta") specifying the diffusion process. See Details.
priors	list of lists containing functions for the prior distributions of the model parameters.
proposals	list of lists containing functions for proposal distributions. Defaults are appropriate in most circumstances.
mcmc_type	Type of MCMC algorithm. Allowable values are "tanner-wong" (Defult) and "fuchs"
alpha	NULL if alpha is to be estimated, else a numeric value of a numeric vector specifying the value of the parameter for all the branches/edges. In the latter case, the values must be specifying in order of the edges in the tr object.
mu	same as for alpha
sigma	same as for alpha
N	data augmentation frequency.
init_method	method for initial data imputation. Currently only the "sim" option is available.
update_method	method for data imputation during the MCMC. The "subtree" will only update a random part of the tree at each iteration, where as the option "tree" will update the whole tree. See Details
iters	number of MCMC iterations.
method	numerical approximation method to use. Default is "euler". "milstein" is also allowed.
fossils	fossil taxa
...	further arguments for future extensions.

Details

Given the root and tip values, the tree, drift and diffusion functions, the Data Augmentation - Markov Chain Monte Carlo (DA-MCMC) estimates the parameters. Parameters may be the same across the tree or allowed to differ in different parts of the tree.

Due to the low frequency nature of the data, we employ Data Augmentation of the evolutionary trajectory (the 'fossil record'), effectively imputing the missing trajectory, and updating trajectories at each MCMC iteration.

If the diffusion process considered is the Ornstein-Uhlenbeck (OU) process, Cox-Ingersoll-Ross (CIR) process or the Beta process (Beta) then the process can be specified by setting the `model` argument to "OU", "CIR", or "Beta", respectively. In the case of a user-defined diffusion process, the estimation of model parameters can be done by specifying the drift and diffusion coefficients in a list assigned to `model`. In this case, the list object `model` must include functions `d` and `s` which are functions of `t`, `x` and `theta` which are the time variable, the trait variable, and a vector consisting of the parameter values `alpha`, `mu` and `sigma`, and the drift coefficient as a list containing a `quote` and diffusion coefficient as a list objects which include `diffusion` as the diffusion coefficient and `x` as the first derivative of diffusion coefficient. See the Examples.

The Ornstein–Uhlenbeck (OU) model is given by

$$dX_t = \alpha(\mu - X_t)dt + \sigma\, dW_t,$$

with $X_0 = x_0 > 0$, W_t is Brownian motion, α, μ, and σ are the model parameters where α and σ are positive values.

The Cox-Ingersoll-Ross (CIR) model is given by

$$dXt = \alpha(\mu - X_t)dt + \sqrt{\epsilon\, X_t}dWt,$$

with $X_0 = x_0 > 0$, where W_t is Brownian motion, α, μ, and ϵ are the model parameters which are all positive values. If `model = "CIR"` is specified, then the parameter estimation is done by using the transformation $Y = \sqrt{(}X)$ of the Ito diffusion process.

The Beta model is given by

$$dX_t = \alpha(\mu - X_t)dt + \sqrt{\epsilon X_t(1 - X_t)}dW_t,$$

with $X_0 = x_0 > 0$, W_t is Brownian motion, α, μ, and ϵ are the model parameters where α and ϵ are positive values. If the `model = "Beta"` is specified, then the parameter estimation is done by using transformation $Y = 2\ sin^{-1}(X)$ of the Itoh diffusion process.

Methods (by class)

- `default`: Bayesian Estimator of Diffusion Processes on Phylogenies

Examples

```
set.seed(1)
rpkgs <- c("sde", "ape", "msm")
lapply(rpkgs, require, character.only = TRUE)
# Number of tips
ntips <- 128
```

```
# SDE parameters
true.alpha <- 10
true.mu <- 5
true.sigma <- 2
t.root.value <- true.mu
iters <- 200
# Generate tip kvalues
set.seed(1)
tr <-  compute.brlen(stree(n=ntips, type="balanced"))
f_TrCir <- function(x, l)
  rcCIR(n=1, Dt=l, x0=x, theta=c(true.alpha*true.mu, true.alpha, true.sigma))
t.tipdata <- rTraitCont(tr, f_TrCir, ancestor = FALSE, root.value = t.root.value)

set.seed(1)
model.1 <- fit_model(tr=tr, tipdata=t.tipdata, rt_value=t.root.value, iters=iters,
                model = "CIR", alpha = 10,  mu = 15, sigma = NULL,
                N=10, init_method = "sim", update_method = "subtree")

# Look at the MCMC trace of the parameters
# summary(model.1)
model.1

## Use the coda package to analyse the mcmc chain
# library(coda)
# plot(model.1$mcmctrace)
# summary(model.1$mcmctrace)

## The same can be done using the user-defined specification:
model <- list()
model$d <- function (t, x, theta) {
  ((theta[1]*theta[2] - 0.25* theta[3]^2) / (2 * x)) - theta[2] * x / 2
}
model$s <- function(t, x, theta) {
  0.5 * theta[3]
}
model$drift <- quote(((alpha * mu - 0.25 * sigma^2) / (2*x)) -
                     alpha * x / 2)
model$diffusion <- quote(sigma/2)
model$dx_diffusion <- quote(0)

tipdata  <- sqrt(t.tipdata)
rt_value <- sqrt(t.root.value)
set.seed(1)
model.2 <- fit_model(tr=tr, tipdata=tipdata, rt_value=rt_value, iters=iters,
                model = model, alpha = 10,  mu = 15, sigma = NULL,
                N=10, init_method = "sim", update_method = "subtree")
```

Description

Starting from the root of the tree - which is assumed to start at time 0, and the root value is known - a recursive scheme is used in simulating a CIR process in the branches of the tree.

Usage

```
phylo_sde(tr, rt_value, N, theta, model, method = "euler", ...)
```

Arguments

tr	a modified object of class phylo as in the ape package. In this version, the CIR process parameters alpha, mu and sigma for each of the branch is included as vectors in the same order as the edge (branch) labelling.
rt_value	value at the root.
N	data imputation frequency.
theta	matrix of parameter values for each edge of the tree.
model	a list containing drift, diffusion and their partial differentiation as quotes. For the Euler scheme the drift coefficient as drift, the diffusion coefficient as diffusion, and the partial differentiation of diffusion by x as dx_diffusion is required. See the Examples.
method	currently only the "euler" scheme is used.
...	not used.

Details

The number of samples imputed along a branch (edge) is proportional to the length of the branch. First, the samples are imputed for the two root edges. The end points of these are taken to be the starting points of the successive branches. This process is done recursively along the tree until the tip nodes are reached.

The number of samples imputed on a branch is equal to round(N * branch_length). As the next branch starts from the end point of the previous, the branch start and stop times can change, which depends on the data imputation frequency N.

In this case, if length of an edge is small, no samples may be imputed for such a branch. As such, the simulated output may contain branches of zero length. This can be avoided by employing higher value for N.

Value

lst a list of length equal to the number of branches in the object tr. Elements of lst are time series objects which are the simulated paths.

Examples

```
set.seed(1)
rpkgs <- c("sde", "ape", "msm")
lapply(rpkgs, require, character.only = TRUE)
# Number of tips
```

```
# Random tree with 64 tips
tr <-  compute.brlen(rtree(n=64))

# SDE parameters
Nedges <- length(tr$edge.length)
dclade <- max(which(tr$edge[,1] == tr$edge[1,1])) - 1
alpha <- mu <- sigma <- rep(0, Nedges)
alpha[1:Nedges]  <- 0.1
mu[1:Nedges] <- 0
sigma[1:Nedges] <- 1
rt_value <- 0
tipdata <- rTraitCont(tr, "OU", sigma=sigma, alpha=alpha, theta=mu,
                      root.value=rt_value)
model <- list()
model$d <- function (t, x, theta) {
  theta[1] * (theta[2] - x)
}
model$s <- function(t, x, theta) {
  theta[3]
}
model$drift <- quote(alpha * (mu - x))
model$diffusion <- quote(sigma)
model$dx_diffusion <- quote(0)
theta <- cbind(alpha=alpha, mu=mu, sigma=sigma)
N <- 100
lst <- phylo_sde (tr=tr, rt_value=rt_value, theta=theta, model=model,
                  N=N, method="euler")
```

`tree_logL` *Calculates the Log Likelihood of the Diffusion Process on a Phylogeny*

Description

Euler approximated Log likelihood of the diffusion process on the tree.

Usage

```
tree_logL(tr, tipdata, lst, alpha, mu, sigma, model, method, fossils,  ...)
```

Arguments

`tr`	object of class "phylo" as in R package ape which contains the information of the tree structure.
`tipdata`	a numeric vector containing tip values in the same order of the tip labels in `tr$tip.label`.
`lst`	list containing the diffusion paths of the `tr` object in the same order as the edges in the `tr` object.
`alpha`	vector containing the parameter value of theta_1 for each edge `tr` object, and the parameters location corresponds to the same edge numbering in the `tr` object.

mu	similar to the definition of alpha.
sigma	similar to the definition of alpha.
model	a list which contains functions d, s and, possibly, s_x which are the drift component, diffusion component, and partial derivative of diffusion component of the diffusion process.
method	Numerical method to approximate the SDE.
fossils	Name of nodes that are fossils.
...	not used.

Value

logL a number.

Examples

```
set.seed(1)
rpkgs <- c("sde", "ape", "msm")
lapply(rpkgs, require, character.only = TRUE)
# Number of tips
# Random tree with 64 tips
tr <-  compute.brlen(rtree(n=64))

# SDE parameters
Nedges <- length(tr$edge.length)
dclade <- max(which(tr$edge[,1] == tr$edge[1,1])) - 1
alpha <- mu <- sigma <- rep(0, Nedges)
alpha[1:Nedges]  <- 0.1
mu[1:Nedges] <- 0
sigma[1:Nedges] <- 1
rt_value <- 0
tipdata <- rTraitCont(tr, "OU", sigma=sigma, alpha=alpha, theta=mu,
                      root.value=rt_value)
model <- list()
model$d <- function (t, x, theta) {
  theta[1] * (theta[2] - x)
}
model$s <- function(t, x, theta) {
  theta[3]
}
model$drift <- quote(alpha * (mu - x))
model$diffusion <- quote(sigma)
model$dx_diffusion <- quote(0)
theta <- cbind(alpha=alpha, mu=mu, sigma=sigma)
N <- 100
lst <- phylo_sde (tr=tr, rt_value=rt_value, theta=theta, model=model,
                  N=N, method="euler")

loglike <-  tree_logL (tr=tr, tipdata=tipdata, lst=lst,
                               alpha=theta[, "alpha"],
                               mu=theta[, "mu"],
```

```
sigma=theta[, "sigma"], model,
method = "euler")
```

Index