Package 'MultiBD'

December 5, 2016

Type Package

Title Multivariate Birth-Death Processes

Version 0.2.0

Date 2016-07-19

Author Lam S.T. Ho [aut, cre],
Marc A. Suchard [aut],
Forrest W. Crawford [aut],
Jason Xu [ctb],
Vladimir N. Minin [ctb]

Maintainer Marc A. Suchard <msuchard@ucla.edu>

Description Computationally efficient functions to provide direct likelihood-based inference for partially-observed multivariate birth-death processes. Such processes range from a simple Yule model to the complex susceptible-infectious-removed model in disease dynamics. Efficient likelihood evaluation facilitates maximum likelihood estimation and Bayesian inference.

License Apache License 2.0

Depends R (>= 3.1.0)

Imports Rcpp (>= 0.11.2), RcppParallel

LinkingTo Rcpp, BH, RcppParallel

Suggests testthat, knitr, rmarkdown, MCMCpack, ggplot2, matrixStats, plotrix

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-12-05 18:28:46

R topics documented:

`bbd_prob` *Transition probabilities of a birth/birth-death process*

Description

Computes the transition pobabilities of a birth/birth-death process using the continued fraction representation of its Laplace transform

Usage

```
bbd_prob(t, a0, b0, lambda1, lambda2, mu2, gamma, A, B, nblocks = 256,
  tol = 1e-12, computeMode = 0, nThreads = 4, maxdepth = 400)
```

Arguments

`t`	time
`a0`	total number of type 1 particles at `t = 0`
`b0`	total number of type 2 particles at `t = 0`
`lambda1`	birth rate of type 1 particles (a two variables function)
`lambda2`	birth rate of type 2 particles (a two variables function)
`mu2`	death rate function of type 2 particles (a two variables function)
`gamma`	transition rate from type 2 particles to type 1 particles (a two variables function)
`A`	upper bound for the total number of type 1 particles
`B`	upper bound for the total number of type 2 particles
`nblocks`	number of blocks
`tol`	tolerance
`computeMode`	computation mode
`nThreads`	number of threads
`maxdepth`	maximum number of iterations for Lentz algorithm

Value

a matrix of the transition probabilities

References

Ho LST et al. 2015. "Birth(death)/birth-death processes and their computable transition probabilities with statistical applications". In review.

Examples

```
## Not run:
data(Eyam)

# (R, I) in the SIR model forms a birth/birth-death process

loglik_sir <- function(param, data) {
  alpha <- exp(param[1]) # Rates must be non-negative
  beta  <- exp(param[2])
  N <- data$S[1] + data$I[1] + data$R[1]

  # Set-up SIR model with (R, I)

  brates1 <- function(a, b) { 0 }
  brates2 <- function(a, b) { beta  * max(N - a - b, 0)  * b }
  drates2 <- function(a, b) { 0 }
  trans21 <- function(a, b) { alpha * b }

  sum(sapply(1:(nrow(data) - 1), # Sum across all time steps k
             function(k) {
               log(
                 bbd_prob( # Compute the transition probability matrix
                   t  = data$time[k + 1] - data$time[k], # Time increment
                   a0 = data$R[k], b0 = data$I[k],       # From: R(t_k), I(t_k)
                   brates1, brates2, drates2, trans21,
                   A = data$R[k + 1], B = data$R[k + 1] + data$I[k] - data$R[k],
                   computeMode = 4, nblocks = 80         # Compute using 4 threads
                 )[data$R[k + 1] - data$R[k] + 1,
                   data$I[k + 1] + 1]                    # To: R(t_(k+1)), I(t_(k+1))
               )
             }))
}

loglik_sir(log(c(3.204, 0.019)), Eyam) # Evaluate at mode

## End(Not run)
```

---

dbd_prob *Transition probabilities of a death/birth-death process*

Description

Computes the transition pobabilities of a death/birth-death process using the continued fraction representation of its Laplace transform

Usage

```
dbd_prob(t, a0, b0, mu1, lambda2, mu2, gamma, a = 0, B, nblocks = 256,
  tol = 1e-12, computeMode = 0, nThreads = 4, maxdepth = 400)
```

Arguments

t	time
a0	total number of type 1 particles at t = 0
b0	total number of type 2 particles at t = 0
mu1	death rate of type 1 particles (a two variables function)
lambda2	birth rate of type 2 particles (a two variables function)
mu2	death rate function of type 2 particles (a two variables function)
gamma	transition rate from type 2 particles to type 1 particles (a two variables function)
a	lower bound for the total number of type 1 particles (default a = 0)
B	upper bound for the total number of type 2 particles
nblocks	number of blocks
tol	tolerance
computeMode	computation mode
nThreads	number of threads
maxdepth	maximum number of iterations for Lentz algorithm

Value

a matrix of the transition probabilities

References

Ho LST et al. 2016. "Birth(death)/birth-death processes and their computable transition probabilities with statistical applications". In review.

Examples

```
## Not run:
data(Eyam)

loglik_sir <- function(param, data) {
  alpha <- exp(param[1]) # Rates must be non-negative
  beta  <- exp(param[2])

  # Set-up SIR model
  drates1 <- function(a, b) { 0 }
  brates2 <- function(a, b) { 0 }
  drates2 <- function(a, b) { alpha * b     }
  trans12 <- function(a, b) { beta  * a * b }

  sum(sapply(1:(nrow(data) - 1), # Sum across all time steps k
             function(k) {
               log(
                 dbd_prob(  # Compute the transition probability matrix
                   t  = data$time[k + 1] - data$time[k], # Time increment
                   a0 = data$S[k], b0 = data$I[k],       # From: S(t_k), I(t_k)
```

```
                  drates1, brates2, drates2, trans12,
                  a = data$S[k + 1], B = data$S[k] + data$I[k] - data$S[k + 1],
                  computeMode = 4, nblocks = 80         # Compute using 4 threads
                )[1, data$I[k + 1] + 1]                 # To: S(t_(k+1)), I(t_(k+1))
              )
            }))
  }

  loglik_sir(log(c(3.204, 0.019)), Eyam) # Evaluate at mode

## End(Not run)

# Birth-death-shift model for transposable elements

lam = 0.0188; mu = 0.0147; v = 0.00268; # birth, death, shift rates

drates1 <- function(a, b) { mu * a }
brates2 <- function(a, b) { lam * (a + b) }
drates2 <- function(a, b) { mu * b }
trans12 <- function(a, b) { v * a }

# Get transition probabilities
p <- dbd_prob(t = 1, a0 = 10, b0 = 0,
              drates1, brates2, drates2, trans12,
              a = 0, B = 50)
```

Eyam *Eyam plague.*

Description

A dataset containing the number of susceptible, infectious and removed individuals during the Eyam plague from June 18 to October 20, 1666.

Usage

```
data(Eyam)
```

Format

A data frame with 8 rows and 4 variables:

time Months past June 18 1666

S Susceptible

I Infectious

R Removed

References

Ragget G (1982). A stochastic model of the Eyam plague. Journal of Applied Statistics 9, 212-226.

`MultiBD` *Multivariate birth-death processes*

Description

The MultiBD package computes the transition probabilities of several multivariate birth-death processes.

References

Ho LST et al. 2016. "Birth(death)/birth-death processes and their computable transition probabilities with statistical applications". In review.

`SIR_prob` *Transition probabilities of an SIR process*

Description

Computes the transition pobabilities of an SIR process using the bivariate birth process representation

Usage

```
SIR_prob(t, alpha, beta, S0, I0, nSI, nIR, direction = c("Forward",
  "Backward"), nblocks = 20, tol = 1e-12, computeMode = 0, nThreads = 4)
```

Arguments

`t`	time
`alpha`	removal rate
`beta`	infection rate
`S0`	initial susceptible population
`I0`	initial infectious population
`nSI`	number of infection events
`nIR`	number of removal events
`direction`	direction of the transition probabilities (either `Forward` or `Backward`)
`nblocks`	number of blocks
`tol`	tolerance
`computeMode`	computation mode
`nThreads`	number of threads

Value

a matrix of the transition probabilities

Examples

```
data(Eyam)

loglik_sir <- function(param, data) {
  alpha <- exp(param[1]) # Rates must be non-negative
  beta  <- exp(param[2])

  if(length(unique(rowSums(data[, c("S", "I", "R")]))) > 1) {
    stop ("Please make sure the data conform with a closed population")
  }

  sum(sapply(1:(nrow(data) - 1), # Sum across all time steps k
             function(k) {
               log(
                 SIR_prob(  # Compute the forward transition probability matrix
                   t  = data$time[k + 1] - data$time[k], # Time increment
                   alpha = alpha, beta = beta,
                   S0 = data$S[k], I0 = data$I[k],       # From: R(t_k), I(t_k)
                   nSI = data$S[k] - data$S[k + 1], nIR = data$R[k + 1] - data$R[k],
                   computeMode = 4, nblocks = 80         # Compute using 4 threads
                 )[data$S[k] - data$S[k + 1] + 1,
                   data$R[k + 1] - data$R[k] + 1]        # To: R(t_(k+1)), I(t_(k+1))
               )
             }))
}

loglik_sir(log(c(3.204, 0.019)), Eyam) # Evaluate at mode
```

Index