Package ‘deBInfer’

November 17, 2022

Type Package
Title Bayesian Inference for Differential Equations
Version 0.4.4
Date 2022-11-14
BugReports https://github.com/pboesu/debinfer/issues
URL https://github.com/pboesu/debinfer
Description A Bayesian framework for parameter inference in differential equations. This approach offers a rigorous methodology for parameter inference as well as modeling the link between unobservable model states and parameters, and observable quantities. Provides templates for the DE model, the observation model and data likelihood, and the model parameters and their prior distributions. A Markov chain Monte Carlo (MCMC) procedure processes these inputs to estimate the posterior distributions of the parameters and any derived quantities, including the model trajectories. Further functionality is provided to facilitate MCMC diagnostics and the visualisation of the posterior distributions of model parameters and trajectories.
License GPL-3
LazyData TRUE
RoxygenNote 7.1.2
Depends R (>= 3.5.0), deSolve
Imports truncdist, coda, RColorBrewer, MASS, stats, mvtnorm, graphics, grDevices, plyr, PBSddesolve, methods
Suggests testthat, knitr, rmarkdown, devtools, R.rsp, beanplot
VignetteBuilder knitr, R.rsp
NeedsCompilation no
Author Philipp H Boersch-Supan [aut, cre]
  (https://orcid.org/0000-0001-6723-6833),
  Leah R Johnson [aut] (https://orcid.org/0000-0002-9922-579X),
  Sadie J Ryan [aut] (https://orcid.org/0000-0002-4308-6321)
Maintainer Philipp H Boersch-Supan <pboesu@gmail.com>
Repository CRAN
Date/Publication 2022-11-17 12:00:24 UTC
**R topics documented:**

- chytrid .................................................. 2
- debinfer_cov ........................................... 3
- debinfer_par .......................................... 3
- dehits .................................................. 5
- depars ................................................... 5
- de_mcmc ................................................ 6
- is.debinfer_parlist .................................. 7
- is.debinfer_result .................................... 7
- logd_prior .............................................. 8
- logistic ................................................. 8
- log_post_params ...................................... 9
- log_prior_params ..................................... 9
- pairs.debinfer_result ................................ 10
- plot.debinfer_result ................................ 10
- plot.post_sim_list .................................... 11
- post_prior_densplot .................................. 11
- post_sim ................................................ 12
- prior_draw_rev ....................................... 13
- propose_joint_rev .................................... 13
- propose_single_rev ................................... 14
- reshape_post_sim ..................................... 14
- setup_debinfer ....................................... 15
- solve_de ................................................. 15
- summary.debinfer_result ............................ 16
- update_sample_rev .................................... 17

### chytrid

*Chytrid fungus data set*

**Description**

Replicated spore counts of an experimental culture of the chytrid fungus *Batrachochytrium dendrobatidis*. This dataset is a subset of the observations from the experimental study conducted by Voyles et al. (2012).

**Format**

A data.frame with 76 rows and two columns

- **time** days since the start of the experiment
- **count** count of zoospores (x 1e4)

**References**

debinfer_cov

Description

debinfer_cov

Usage

debinfer_cov(var.names, sigma = diag(length(names)), name, samp.type = "rw")

Arguments

var.names  names of the parameters that are to be proposed together
sigma  covariance matrix
name  name of the joint block
samp.type  character; type of sampler. currently only "rw" = Normal random walk is implemented for multivariate proposals

Value

a debinfer_cov object

debinfer_par

Description

Creates an object containing all the necessary bits for a parameter i.e. initial values, prior distributions, hyper-parameters, tuning parameters, etc. to set up a debinfer analysis

Usage

debinfer_par(
  name,
  var.type,
  fixed,
  value,
  joint = NULL,
  prior = NULL,
  hypers = NULL,
  prop.var = NULL,
  samp.type = NULL
)
**Arguments**

- **name**: character vector; name of the variable
- **var.type**: character vector; type of the variable "de" = parameter for the differential equation, "obs" = parameter of the observation model, "init" = initial condition for a state variable in the differential equation
- **fixed**: boolean; TRUE = parameter is taken to be fixed, FALSE = parameter is to be estimated by MCMC
- **value**: numeric; parameter value. For fixed parameters this is the value used in the analysis for free parameters this is the starting value used when setting up the MCMC chain
- **joint**: integer; number of block for joint proposal; NULL means the parameter is not to be jointly proposed
- **prior**: character; name of the probability distribution for the prior on the parameter. Must conform to standard R naming of d/r function pairs, e.g. beta (foo = beta), binomial binom, Cauchy cauchy, chi-squared chisq, exponential exp, Fisher F f, gamma gamma, geometric geom, hypergeometric hyper, logistic logis, lognormal lnorm, negative binomial nbinom, normal norm, Poisson pois, Student t t, uniform unif, Weibull weibull. Priors from the truncdist package are available by default. User priors can be provided but must be available in the environment from which de_mcmc is called.
- **hypers**: list of numeric vectors, hyperparameters for the prior; mean only for mvnorm. Can include trunc for truncated pdfs from package truncdist.
- **prop.var**: numeric; tuning parameters. For Normal proposals ('samp.type="rw"' or 'samp.type="rw-ref"'), this must be a positive number representing the standard deviation of the proposal distribution for each parameter. For the asymmetric uniform proposal distribution ('samp.type="rw-unif"') two positive numeric values are required and the proposal will then have the bounds \( \text{prop.var}[1]/\text{prop.var}[2]*\text{current_proposal} \) and \( \text{prop.var}[2]/\text{prop.var}[1]*\text{current_proposal} \). See Boersch-Supan et al. (2016).
- **samp.type**: character; type of sampler: "rw" = Normal random walk, "ind" = independence, "rw-unif" = asymmetric uniform distribution, "rw-ref" = reflecting random walk sampler on the bounds of the prior support (cf. Hoff 2009, Chapter 10.5.1; Yang and Rodriguez 2013)

**Value**

returns an object of class debinfer_par to be fed to the mcmc setup function

**References**

- Hoff 2009, A First Course in Bayesian Statistical Methods, Springer
deinits  
Get starting/fixed values of DE initial values

**Description**
Accessor function for initial values

**Usage**
```r
deinits(x)
```

**Arguments**
- `x`  
a debinfer_result or debinfer_parlist object

**Value**
a named numeric vector

depars  
Get starting/fixed values of DE parameters

**Description**
Accessor function for parameters

**Usage**
```r
depars(x)
```

**Arguments**
- `x`  
a debinfer_result or debinfer_parlist object

**Value**
a named numeric vector
Description

Bayesian inference for a deterministic DE model (with models solved via an DE solver) with an observation model.

Usage

de_mcmc(
  N,
  data,
  de.model,
  obs.model,
  all.params,
  ref.params = NULL,
  ref.inits = NULL,
  Tmax,
  data.times,
  cnt = 10,
  plot = TRUE,
  sizestep = 0.01,
  solver = "ode",
  verbose.mcmc = TRUE,
  verbose = FALSE,
  ...
)

Arguments

N integer, number of MCMC iterations
data data.frame of time course observations to fit the model to. The observations must be ordered ascending by time.
de.model a function defining a DE model, compliant with the solvers in deSolve or PBSddesolve
obs.model a function defining an observation model. Must be a function with arguments 'data', 'sim.data', 'samp'.
all.params debinfer_parlist containing all model, MCMC, and observation
ref.params an optional named vector containing a set of reference parameters, e.g. the true parameters underlying a simulated data set
ref.inits an optional named vector containing a set of reference initial values, e.g. the true initial values underlying a simulated data set
Tmax maximum timestep for solver
data.times time points for which observations are available
is.debinfer_parlist

Cnt integer interval at which to print and possibly plot information on the current state of the MCMC chain
plot logical, plot traces for all parameters at the interval defined by cnt
sizestep timestep for solver to return values at, only used if data.times is missing
solver the solver to use. 1 or "ode" = deSolve::ode; 2 or "dde" = PBSddesolve::dde; 3 or "dede" = deSolve::dede
verbose.mcmc logical display MCMC progress messages
verbose logical display verbose solver output
... further arguments to the solver

Value
a debinfer_result object containing input parameters, data and MCMC samples

is.debinfer_parlist

Description
Check debinfer_parlist class

Usage
is.debinfer_parlist(x)

Arguments
x an object

is.debinfer_result

Description
Check debinfer_result class

Usage
is.debinfer_result(x)

Arguments
x an object
Description
Evaluates the log probability density of value given a name of a prior pdf and the corresponding hyperparameters

Usage
\[ \text{logd\_prior}(x, \text{pdf}, \text{hypers}) \]

Arguments
- \( x \): numeric; vector of values.
- \( \text{pdf} \): character; name of a probability function. Must conform to base R nomenclature of \( d/r \) function pairs. Can include \text{trunc} for truncated pdfs from package \text{truncdist}.
- \( \text{hypers} \): list; a list of parameters to be passed to the density function.

Value
the value of the log density function evaluated at \( x \)

logistic

Logistic growth data set

Description
Simulated data from the logistic growth model with \( N_0=0.1, r=0.1 \) and \( K=10 \)

Format
A data.frame with 36 rows and 3 columns
- \text{time} \quad \text{time since start of the model}
- \text{N\_true} \quad \text{Numerical solution of } N_t
- \text{N\_noisy} \quad N_t \text{ with the addition of log-normal noise, where } sdlog = 0.05
**log_post_params**

**Description**

evaluate the likelihood given the data, the current deterministic model solution and the observation model

**Usage**

```r
log_post_params(samp, w.p, data, pdfs, hyper, sim.data, obs.model)
```

**Arguments**

- `samp`: named numeric; current sample
- `w.p`: character; parameter names
- `data`: data
- `pdfs`: character, prior pdf names
- `hyper`: list, hyper parameters for the priors
- `sim.data`: solver output
- `obs.model`: function containing the observation model

**log_prior_params**

**Description**

evaluate prior density at current parameter values

**Usage**

```r
log_prior_params(samp, pdfs, w.p, hyper)
```

**Arguments**

- `samp`: named numeric; current sample
- `pdfs`: character, prior pdf names
- `w.p`: character; parameter names
- `hyper`: list of named hyper parameters for the priors
pairs.debinfer_result  
Pairwise posterior marginals

Description
Plots pairwise correlations of posterior marginals

Usage
## S3 method for class 'debinfer_result'
pairs(x, trend = FALSE, scatter = FALSE, burnin = NULL, medians = TRUE, ...)

Arguments
- **x**: a deBInfer_result object
- **trend**: logical, add loess smooth
- **scatter**: logical, add scatterplot of posterior samples
- **burnin**: integer, number of samples to discard from start of chain before plotting
- **medians**: logical, plot marginal medians on contour plot
- **...**: further arguments to plot.default (the call that draws the scatter/contour plot)

plot.debinfer_result  
Plot inference outputs

Description
Plots the inference results from a debinfer_result object

Usage
## S3 method for class 'debinfer_result'
plot(x, plot.type = "coda", burnin = 1, ...)

Arguments
- **x**: a deBInfer_result object
- **plot.type**: character, which type of plot. Options are "coda" for coda::plot.mcmc, "post_prior" for deBInfer::post_prior_densplot.
- **burnin**: numeric, number of samples to discard before plotting
- **...**: further arguments to methods

See Also
post_prior_densplot, plot.mcmc, pairs.debinfer_result
plot.post_sim_list       Plot posterior trajectory

Description
Plots the inference results from a debinfer_result object

Usage
## S3 method for class 'post_sim_list'
plot(
  x,
  plot.type = "medianHDI",
  col = c("red", "darkgrey"),
  lty = c(1, 2),
  auto.layout = TRUE,
  panel.first = NULL,
  ...
)

Arguments
  x       a post_sim or post_sim_list object
  plot.type character, which type of plot. Options are "ensemble" and "medianHDI".
  col     color, for plot.type = "medianHDI" the first element is used for the median, the second for the HDI
  lty     line type, for plot.type = "medianHDI" the first element is used for the median, the second for the HDI
  auto.layout logical, should the layout for plot.type = "medianHDI" be determined automatically?
  panel.first an expression to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for adding data points.
  ...
    further arguments to methods

post_prior_densplot      Plot posterior marginals and corresponding priors

Description
Plots posterior densities and the densities of the corresponding priors. The prior density is automatically evaluated for the range given by the x-axis limits of the plot (which defaults to the posterior support).
Usage

post_prior_densplot(
  result,
  param = "all",
  burnin = NULL,
  prior.col = "red",
  n = 1000,
  ...
)

Arguments

result            a deBInfer_result object
param              character, name of parameter to plot. "all" (default) plots all parameters
burnin            numeric, number of samples to discard before plotting
prior.col          character color for prior density
n,                 integer, number of points at which to evaluate the prior density.
...                further arguments to coda::densplot

Description

post_sim

Usage

post_sim(x, n = 100, times, output = "all", burnin = NULL, prob = 0.95, ...)

Arguments

x                   debinfer_result object
n                   number of simulations
times               numeric a vector of times at which the ODE is to be evaluated. Defaults to NULL.
output              character, "sims", "all", "HDI"
burnin              integer, number of samples to discard from the start of the mcmc chain
prob                A numeric scalar in the interval (0,1) giving the target probability content of the intervals. The nominal probability content of the intervals is the multiple of 1/nrow(obj) nearest to prob.
...                  additional arguments to solver

Value

a post_sim object containing a list of de solutions or summaries thereof
**prior_draw_rev**

**Description**

draw from prior

**Usage**

prior_draw_rev(b, hypers, prior.pdf)

**Arguments**

- **b**: current value of a parameter
- **hypers**: list of hyper parameters, named appropriately for the corresponding prior.pdf
- **prior.pdf**: string name of probability distribution following base R conventions, or those of additionally loaded packages

**propose_joint_rev**

**Description**

joint proposal function

**Usage**

propose_joint_rev(samps, s.ps, cov.mat)

**Arguments**

- **samps**: current sample of the MCMC chain
- **s.ps**: debinfer_parlist object representing the parameters that are to be proposed
- **cov.mat**: debinfer_cov object; covariance matrix for the proposal

**Details**

Function to jointly propose parameters using a multivariate normal proposal distribution
propose_single_rev

Description
propose a parameter individually

Usage
propose_single_rev(samps, s.p)

Arguments
samps current sample of the MCMC chain
s.p debinfer_par object representing the parameter that is to be proposed

reshape_post_sim

Description
Take a list of DE model solutions and transform into a list of of matrices, one for each state variable, where each row is an iteration, and each column is a time point

Usage
reshape_post_sim(x)

Arguments
x a post_sim object
**Description**

Creates an object of class debinfer_parlist containing initial values, parameters, prior distributions, hyperparameters tuning parameters etc. to set up a debinfer analysis.

**Usage**

```r
setup_debinfer(...)
```

**Arguments**

```r
...
```

Debinfer_par objects to be combined into a debinfer_parlist.

**Value**

returns an S3 object of class debinfer_parlist to be fed to the mcmc function.

---

**Description**

solve_de

**Usage**

```r
solve_de(
  sim,
  params,
  inits,
  Tmax,
  numsteps = 10000,
  solver = "ode",
  sizestep = NULL,
  verbose = FALSE,
  data.times = NULL,
  method = "lsoda",
  ...
)
```
Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim</td>
<td>function; solver compatible specification of the DE</td>
</tr>
<tr>
<td>params</td>
<td>numeric; named vector of parameter values</td>
</tr>
<tr>
<td>inits</td>
<td>numeric; initial values. Must be in the same order as specified within sim!</td>
</tr>
<tr>
<td>Tmax</td>
<td>numeric; maximum timestep</td>
</tr>
<tr>
<td>numsteps</td>
<td>numeric</td>
</tr>
<tr>
<td>solver</td>
<td>Choice of solver to use 1 or &quot;ode&quot; = deSolve::ode, 2 or &quot;dde&quot; = PBSdodesolve::dde, 3 or &quot;dede&quot; = deSolve::dede</td>
</tr>
<tr>
<td>sizestep</td>
<td>for solver</td>
</tr>
<tr>
<td>verbose</td>
<td>passed to deSolve::ode</td>
</tr>
<tr>
<td>data.times</td>
<td>numeric a vector of times at which the ODE is to be evaluated. Defaults to NULL. If value is supplied it takes precedence over any value supplied to numsteps or sizesteps.</td>
</tr>
<tr>
<td>method</td>
<td>solver method</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments to solver</td>
</tr>
</tbody>
</table>

Value

integrated ode object. Data structure depends on the employed solver.

Description

A wrapper for coda::summary.mcmc

Usage

```r
## S3 method for class 'debinfer_result'
summary(object, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>a deBInfer_result object</td>
</tr>
<tr>
<td>...</td>
<td>further arguments to summary.mcmc</td>
</tr>
</tbody>
</table>

See Also

summary.mcmc
Description
This is the workhorse of the MCMC algorithm

Usage

\[
\text{update\_sample\_rev}(\text{samps, samp.p, cov.mats, data, sim, out, Tmax, sizestep, data.times, l, solver, i, cnt, obs.model, pdfs, hyper, w.p, verbose.mcmc, verbose, is.de, is.single, joint.blocks, ...})
\]

Arguments

- `samps`: row vector of samples from the previous mcmc iteration
- `samp.p`: the parlist created by setup_debinfer
- `cov.mats`: the covariance matrices
- `data`: the observation
- `sim`: the de.model
- `out`: list containing the initial or previous update i.e. list(s=samps[i-1,], inits=inits, p=params, sim.old=sim.start)
- `Tmax`: maximum timestep for solver
sizestep sizestep for solver when not using data.times
data.times times with observations
l number of parameters to be proposed
solver solver choice
i current MCMC iteration
cnt interval for printing/plotting information on chains
obs.model function containing obs model
pdfs names of prior pdfs
hyper list of hyperparameters
w.p names of free parameters
verbose.mcmc logical, print MCMC progress messages
verbose logical, print additional information from solver
is.de logical, parameter is an input for the solver
is.single parameter is to be proposed individually
joint.blocks names of joint blocks
... further arguments to solver
Index

* data
  chytrid, 2
  logistic, 8

chytrid, 2

de_mcmc, 6
debinfer_cov, 3
debinfer_par, 3
deinit, 5
depars, 5

is.debinfer_parlist, 7
is.debinfer_result, 7

log_post_params, 9
log_prior_params, 9
logd_prior, 8
logistic, 8

pairs.debinfer_result, 10, 10
plot.debinfer_result, 10
plot.mcmc, 10
plot.post_sim_list, 11
post_prior_densplot, 10, 11
post_sim, 12
prior_draw_rev, 13
propose_joint_rev, 13
propose_single_rev, 14

reshape_post_sim, 14

setup_debinfer, 15
solve_de, 15
summary.debinfer_result, 16
summary.mcmc, 16

update_sample_rev, 17