Package 'pmwg'

May 9, 2026

Title Particle Metropolis Within Gibbs

Version 0.2.7

Description Provides an R implementation of the Particle Metropolis within Gibbs sampler for model parameter, covariance matrix and random effect estimation. A more general implementation of the sampler based on the paper by Gunawan, D., Hawkins, G. E., Tran, M. N., Kohn, R., & Brown, S. D. (2020) <doi:10.1016/j.jmp.2020.102368>. An HTML tutorial document describing the package is available at <https://university-of-newcastle-research.github.io/samplerDoc/> and includes several detailed examples, some background and troubleshooting steps.

License GPL-3

URL https://github.com/university-of-newcastle-research/pmwg

BugReports https://github.com/university-of-newcastle-research/pmwg/issues

Depends R (>= 3.6.0)

Imports checkmate, coda, condMVNorm, MASS, mvtnorm, stats

Suggests covr, rtdists, testthat, knitr, rmarkdown, V8

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Gavin Cooper [aut, cre, trl] (Package creator and maintainer),
Reilly Innes [aut],
Caroline Kuhne [aut],
Jon-Paul Cavallaro [aut],
David Gunawan [aut] (Author of original MATLAB code),
Guy Hawkins [aut],
Scott Brown [aut, trl] (Original translation from MATLAB to R),
Niek Stevenson [aut]

Maintainer Gavin Cooper <gavin@gavincooper.net>

Repository CRAN
Date/Publication 2024-01-31 05:00:02 UTC

Contents

`as_mcmc` *Return a CODA mcmc object with the required samples*

Description

Given a sampler object and a specification of the samples required, return either an individual coda mcmc object, or a list of mcmc objects.

Usage

```
as_mcmc(sampler, selection = "theta_mu", filter = stages)
```

Arguments

`sampler`	The pmwgs object containing samples to extract.
`selection`	The selection of sample types to return.
`filter`	A filter that defines which stage to draw samples from.

Value

An mcmc object or list containing the selected samples.

Selecting sample types

The values that can be chosen for the `selection` argument can be one of the following list:

`"theta_mu"` the model parameter estimate samples

`"theta_sig"` the covariance matrix estimates, returns a list of mcmc objects, one for each model parameter.

`"alpha"` the random effect estimates, returns a list of mcmc objects, one for each subject.

The default value for `selection` is `"theta_mu"`

Filtering samples

The `filter` argument can take one of two forms:

- An integer vector, usually a sequence of integers, that must fall within the range 1:end.
- A character vector, where each element corresponds to a stage of the sampling process, i.e. one or more of "init", "burn", "adapt" or "sample".

The default value for `filter` is all stages.

Examples

```
par_estimates <- as_mcmc(sampled_forstmann)
par_estimates_sample_stage <- as_mcmc(sampled_forstmann, filter = "sample")
rand_eff <- as_mcmc(
  sampled_forstmann,
  selection = "alpha",
  filter = "sample"
)
```

`augment_sampler_epsilon` *Augment existing sampler object to have subject specific epsilon storage*

Description

Older sampler object will be missing subject specific scaling parameter (epsilon) storage, and running a stage with an updated pmwg will fail. To fix this you can run the augment_sampler_epsilon function to fill the appropriate array internals with NA values

Usage

```
augment_sampler_epsilon(sampler)
```

Arguments

`sampler`	The sampler object to augment

Value

A pmwgs sampler with epsilon array set internally

forstmann *Forstmann et al.'s data*

Description

A dataset containing the speed or accuracy manipulation for a Random Dot Motion experiment.

Usage

```
forstmann
```

Format

A data frame with 15818 rows and 5 variables:

subject integer ID for each subject

rt reaction time for each trial as a double

condition Factor with 3 levels for Speed, Accuracy and Neutral

stim Factor with 2 levels for Left and Right trials

resp Factor with 2 levels for Left and Right responses

Details

Details on the dataset can be found in the following paper:

Striatum and pre-SMA facilitate decision-making under time pressure

Birte U. Forstmann, Gilles Dutilh, Scott Brown, Jane Neumann, D. Yves von Cramon, K. Richard Ridderinkhof, Eric-Jan Wagenmakers.

Proceedings of the National Academy of Sciences Nov 2008, 105 (45) 17538-17542; DOI: 10.1073/pnas.0805903105

Source

https://www.pnas.org/content/105/45/17538

init *Initialise values for the random effects*

Description

Initialise the random effects for each subject using MCMC.

Usage

```
init(
  pmwgs,
  start_mu = NULL,
  start_sig = NULL,
  display_progress = TRUE,
  particles = 100
)
```

Arguments

pmwgs	The sampler object that provides the parameters.
start_mu	An array of starting values for the group means
start_sig	An array of starting values for the group covariance matrix
display_progress	Display a progress bar during sampling
particles	The number of particles to generate in initialisation

Details

Before sampling can start the Particle Metropolis within Gibbs sampler needs initial values for the random effects. The `init` function generates these values using a Monte Carlo algorithm. One alternative methods would be setting the initial values randomly.

Optionally takes starting values for the model parameters and the variance / covariance matrix. All arrays must match the appropriate shape.

For example, with 5 parameters and 10 subjects, the model parameter start means must be a vector of length 5 and the covariance matrix must be an array of 5 x 5.

If the start_mu and start_sig arguments are left at the default (NULL) then start_mu will be sampled from a normal distribution with mean as the prior mean for eac variable and sd as the square of the variance from the prior covariance matrix. start_sig by default is sampled from an inverse wishart (IW) distribution. For a model with the number of parameters N the degrees of freedom of the IW distribution is set to N*3 and the scale matrix is the identity matrix of size NxN.

Value

The sampler object but with initial values set for `theta_mu`, `theta_sig`, `alpha` and other values for the first sample.

Examples

```
lba_ll <- function(x, data) {
  x <- exp(x)
  if (any(data$rt < x["t0"])) {
    return(-1e10)
  }
  sum(
    log(
```

```
      rtdists::dLBA(
        rt = data$rt,
        response = data$correct,
        A = x["A"],
        b = x["A"] + x[c("b1", "b2", "b3")][data$condition],
        t0 = x["t0"],
        mean_v = x[c("v1", "v2")],
        sd_v = c(1, 1),
        silent = TRUE
      )
    )
  )
}
sampler <- pmwgs(
  forstmann,
  c("b1", "b2", "b3", "A", "v1", "v2", "t0"),
  lba_ll
)
sampler <- init(sampler, particles=10)
```

`is.pmwgs` *Test whether object is a pmwgs*

Description

Checks whether object is a Particle Metropolis with Gibbs sampler

Usage

```
is.pmwgs(x)
```

Arguments

`x`	An object to test

Value

logical, whether object inherits from pmwgs

Examples

```
if (is.pmwgs(sampled_forstmann)) {
  print("sampled_forstmann object is a pmwgs")
}
```

pmwgs *Create a PMwG sampler and return the created object*

Description

This function takes a few necessary elements for creating a PMwG sampler. Each pmwgs object is required to have a dataset, a list of parameter names, a log likelihood function and optionally a prior for the model parameters.

Usage

```
pmwgs(data, pars, ll_func, prior = NULL)
```

Arguments

data	A data frame containing empirical data to be modelled. Assumed to contain at least one column called subject whose elements are unique identifiers for each subject. Can be any of data.frame, data.table or tibble, or any other data frame like object that can have subsets created in an identical way.
pars	The list of parameter names to be used in the model
ll_func	A log likelihood function that given a list of parameter values and a data frame (or other data store) containing subject data will return the log likelihood of data given x.
prior	Specification of the prior distribution for the model parameters. It should be a list with two elements, theta_mu_mean and theta_mu_var which fully specify the prior distribution. If left as the default (NULL) then the theta_mu_mean will be all zeroes and theta_mu_var will be 1 on the diagonal and zero elsewhere.

Value

A pmwgs object that is ready to be initialised and sampled.

Examples

```
# Specify the log likelihood function
lba_loglike <- function(x, data) {
  x <- exp(x)
  if (any(data$rt < x["t0"])) {
    return(-1e10)
  }
  # This is faster than "paste".
  bs <- x["A"] + x[c("b1", "b2", "b3")][data$condition]

  out <- rtdists::dLBA(
    rt = data$rt, # nolint
    response = data$stim,
    A = x["A"],
```

```
    b = bs,
    t0 = x["t0"],
    mean_v = x[c("v1", "v2")],
    sd_v = c(1, 1),
    distribution = "norm",
    silent = TRUE
  )
  bad <- (out < 1e-10) | (!is.finite(out))
  out[bad] <- 1e-10
  out <- sum(log(out))
  out
}

# Specify parameter names and priors
pars <- c("b1", "b2", "b3", "A", "v1", "v2", "t0")
priors <- list(
  theta_mu_mean = rep(0, length(pars)),
  theta_mu_var = diag(rep(1, length(pars)))
)

# Create the Particle Metropolis within Gibbs sampler object
sampler <- pmwgs(
  data = forstmann,
  pars = pars,
  ll_func = lba_loglike,
  prior = priors
)

sampler = init(sampler, particles=10)
sampler = run_stage(sampler, stage="burn", iter=10, particles=10)
```

`relabel_samples` *Relabel requested burn-in samples as adaptation*

Description

Given a sampler object and a vector of sample indices, relabel the given samples to be adaptation samples. This will allow them to be used in the calculation of the conditional distribution for efficient sampling.

Usage

```
relabel_samples(sampler, indices, from = "burn", to = "adapt")
```

Arguments

`sampler`	The pmwgs object that we are relabelling samples from
`indices`	The sample iterations from burn-in to relabel
`from`	The stage you want to re-label from. Default is "burn"
`to`	The stage you want to relabel to. Default is "adapt"

Value

The pmwgs object with re-labelled samples

Further information

This should not usually be needed, however if you have a model that is slow to fit, and upon visual inspection and/or trace analysis you determine that during burn-in the samples had already approached the posterior distribution then you can use this function to re-label samples from that point onwards to be classed as adaptation samples.

This will allow them to be used in tests that check for the number of unique samples, and in the building of the conditional distribution (which is used for efficient sampling)

If all old samples do not match 'from' then an error will be raised.

Examples

```
new_pmwgs <- relabel_samples(sampled_forstmann, 17:21)
```

---

run_stage *Run a stage of the PMwG sampler*

Description

Run one of burnin, adaptation or sampling phase from the PMwG sampler. Each stage involves slightly different processes, so for the full PMwG sampling we need to run this three times.

Usage

```
run_stage(
  pmwgs,
  stage,
  iter = 1000,
  particles = 100,
  display_progress = TRUE,
  n_cores = 1,
  n_unique = ifelse(stage == "adapt", 100, NA),
  epsilon = NULL,
  p_accept = 0.8,
  mix = NULL,
  pdist_update_n = ifelse(stage == "sample", 50, NA)
)
```

Arguments

pmwgs	A Particle Metropolis within Gibbs sampler which has been set up and initialised
stage	The sampling stage to run. Must be one of 'burn', 'adapt' or 'sample'.

iter	The number of iterations to run for the sampler. For 'burn' and 'sample' all iterations will run. However for 'adapt' if all subjects have enough unique samples to create the conditional distribution then the stage will finish early.
particles	The default here is 1000 particles to be generated for each iteration, however during the sample phase this should be reduced.
display_progress	Display a progress bar during sampling.
n_cores	Set to more than 1 to use mclapply. Setting n_cores greater than 1 is only permitted on Linux and Mac OS X machines.
n_unique	A number representing the number of unique samples to check for on each iteration of the sampler (An initial test for the generation of the proposal distribution). Only used during the 'adapt' stage.
epsilon	A value between 0 and 1 that controls the extent to which the covariance matrix is scaled when generating particles from the previous random effect. The default will be chosen based on the number of random effects in the model.
p_accept	A value between 0 and 1 that will flexibly tune epsilon to achieve an acceptance ratio close to what you set p_accept to. The default is set at 0.8.
mix	A vector of floats that controls the mixture of different sources for particles. The function numbers_from_proportion is passed this value and includes extra details on what is accepted.
pdist_update_n	The number of iterations in the sample stage after which the proposal distribution will be recomputed.

Details

The **burnin** stage by default selects 50 parameter sample (selected through a Gibbs step) and 50 the previous random effect of each subject. It assesses each particle with the log-likelihood function and samples from all particles weighted by their log-likelihood.

The **adaptation** stage selects and assesses particle in the same was as burnin, however on each iteration it also checks whether each subject has enough unique random effect samples to attempt to create a conditional distribution for efficient sampling in the next stage. If the attempt at creating a conditional distribution fails, then the number of unique samples is increased and sampling continues. If the attempt succeeds then the stage is finished early.

The **final** stage (sampling) by default samples predominantly from the conditional distribution created at the end of adaptation. This is more efficient and allows the number of particles to be reduced whilst still getting a high enough acceptance rate of new samples.

Once complete each stage will return a sampler object with the new samples stored within it.

The progress bar (which is displayed by default) shows the number of iterations out of those requested which have been completed. It also contains additional information at the end about the number of newly generated particles that have been accepted. This is show as New(XXX average across subjects of newly sampled random effects accept rate. See accept_rate for more detail on getting individual accept rate values per subject.

Value

A pmwgs object with the newly generated samples in place.

Examples

```
library(rtdists)
sampled_forstmann$data <- forstmann
run_stage(sampled_forstmann, "sample", iter = 1, particles = 10)
```

`sampled_forstmann` *A sampled object of a model of the Forstmann dataset*

Description

A pmwgs object with a limited number of samples of the Forstmann dataset.

Usage

```
sampled_forstmann
```

Format

A pmwgs object minus the data. A pmwgs opbject is a list with a specific structure and elements, as outlined below.

par_names A character vector containing the model parameter names

n_pars The number of parameters in the model

n_subjects The number of unique subject ID's in the data

subjects A vector containing the unique subject ID's

prior A list that holds the prior for `theta_mu` (the model parameters). Contains the mean (`theta_mu_mean`), covariance matrix (`theta_mu_var`) and inverse covariance matrix (`theta_mu_invar`)

ll_func The log likielihood function used by pmwg for model estimation

samples A list with defined structure containing the samples, see the Samples Element section for more detail

Details

The pmwgs object is missing one aspect, the pmwgs$data element. In order to fully replicate the full object (ie to run more sampling stages) you will need to add the data back in, via sampled_forstmann$data <- forstmann

Samples Element

The samples element of a PMwG object contains the different types of samples estimated by PMwG. These include the three main types of samples `theta_mu`, `theta_sig` and `alpha` as well as a number of other items which are detailed here.

theta_mu samples used for estimating the model parameters (group level), an array of size (n_pars x n_samples)

theta_sig samples used for estimating the parameter covariance matrix, an array of size (n_pars x n_pars x n_samples)

alpha samples used for estimating the subject random effects, an array of size (n_pars x n_subjects x n_samples)

stage A vector containing what PMwG stage each sample was drawn in

subj_ll The winning particles log-likelihood for each subject and sample

a_half Mixing weights used during the Gibbs step when creating a new sample for the covariance matrix

last_theta_sig_inv The inverse of the last samples covariance matrix

idx The index of the last sample drawn

Source

https://www.pnas.org/content/105/45/17538

Index