Package ‘rjpdmp’

February 22, 2022

Type Package

Title Reversible Jump PDMP Samplers

Version 2.0.0

Author Matt Sutton, Augustin Chevalier, Paul Fearnhead, with PolyaGamma simulation code contributed from Jesse Windle and James G. Scott (<https://github.com/jgscott/helloPG>)

Maintainer Matt Sutton <matt.sutton.stat@gmail.com>

Description Provides an implementation of the reversible jump piecewise deterministic Markov processes (PDMPs) methods developed in the paper Reversible Jump PDMP Samplers for Variable Selection (Chevallier, Fearnhead, Sutton 2020, <arXiv:2010.11771>). It also contains an implementation of a Gibbs sampler for variable selection in Logistic regression based on Polya-Gamma augmentation.

License GPL (>= 2)

RoxygenNote 7.1.1

Encoding UTF-8

Imports data.table, Rcpp (>= 0.12.3)

Suggests MASS

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-02-21 23:10:02 UTC

R topics documented:

`rjpdmp-package` *rjpdmp-package*

Description

Implements various reversible jump piecewise-deterministic Markov Process methods including the ZigZag and Bouncy Particle Sampler with Normal or Spherical velocity distributions (Chevallier, Fearnhead, Sutton 2020, https://arxiv.org/abs/2010.11771).

The package can be used to Generates PDMP trajectories for reversible jump

- `zigzag_logit`: ZigZag on logistic likelihood problem
- `zigzag_logit_ss`: ZigZag with subsampling on Logistic likelihood problem
- `bps_s_logit`: BPS with velocities distributed uniformly on the sphere for a Logistic likelihood problem
- `bps_n_logit`: BPS with velocities distributed Normally for a Logistic likelihood problem
- `zigzag_rr`: ZigZag on a robust regression likelihood problem
- `bps_s_rr`: BPS with velocities distributed uniformly on the sphere for a robust regression likelihood problem
- `bps_n_rr`: BPS with velocities distributed Normally for a robust regression likelihood problem

Additional functions

Additional functions for plotting, generating samples, calculating posterior means or probabilities of inclusion

- `plot_pdmp`: Plot marginal densities and joint pairs plots for trajectories and samples of PDMP samplers and optionally MCMC samples for comparison.
- `plot_pdmp_multiple`: Plots to compare PDMP samplers and optionally MCMC samples.
- `gen_sample`: Get samples from PDMP trajectories taking a fixed time discretisation.
- `model_probabilities`: Calculate either marginal probabilities of inclusions or posterior probabilities of specific models.
- `models_visited`: Count the number of times a model is visited
- `marginal_mean`: Calculate the marginal mean using PDMP trajectories

- cond_mean: Calculate the mean conditioned on being in a specific model

Extensions to the package are planned.

---

bps_n_logit *bps_n_logit*

Description

Applies the Reversible Jump BPS Sampler with Velocities distributed from the Normal distribution to a Logistic regression with dirac spike and slab distribution, as detailed in Reversible Jump PDMP Samplers for Variable Selection, 2020. For included variables an independent Gaussian prior is assumed with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
bps_n_logit(
  maxTime,
  dataX,
  datay,
  prior_sigma2,
  x0,
  theta0,
  ref = 0.1,
  rj_val = 0.6,
  ppi = 0.5,
  nmax = 1000000L,
  burn = -1L
)
```

Arguments

`maxTime`	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
`dataX`	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
`datay`	Vector of n observations of a 0, 1-valued variable y.
`prior_sigma2`	Double for the prior variance for included variables.
`x0`	Initial position of the regression parameter
`theta0`	Initial velocity for the sampler (Default has 1s on all components). This should be chosen with unit velocities on each component (regardless of sign).
`ref`	Double for the refreshment rate of the BPS.
`rj_val`	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.

ppi	Double for the prior probability of inclusion (ppi) for each parameter.
nmax	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.
burn	Optional number of iterations to use for burnin. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switchs velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p
## Not run:
bps_fit <- bps_n_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                          prior_sigma2 = 10, theta0 = rep(0, p),
                          x0 = rep(0, p), ref = 0.1, rj_val = 0.6,
                          ppi = ppi, nmax = 1e6, burn = -1)

gibbs_fit <- gibbs_logit(maxTime = 1, dataX = data$dataX, datay =data$dataY,
                         prior_sigma2 = 10,beta = rep(0,p), gamma =rep(0,p),
                         ppi = ppi)

plot_pdmp(bps_fit, coords = 1:2, inds = 1:1e3,burn = .1, nsamples = 1e4,
          mcmc_samples = t(gibbs_fit$beta*gibbs_fit$gamma))

## End(Not run)
```

bps_n_rr *bps_n_rr*

Description

Applies the Reversible Jump BPS Sampler with Velocities drawn from the Normal distribution to a Robust Regression problem with dirac spike and slab prior. Included variables are given an independent Gaussian prior with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
bps_n_rr(
  maxTime,
  dataX,
  datay,
  prior_sigma2,
  x0,
  theta0,
  ref = 0.1,
  rj_val = 0.5,
  ppi = 0.5,
  nmax = 1000000L,
  burn = -1L
)
```

Arguments

`maxTime`	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
`dataX`	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
`datay`	Vector of n observations of a continuous response variable y.
`prior_sigma2`	Double for the prior variance for included variables.
`x0`	Initial position of the regression parameter
`theta0`	Initial velocity for the sampler.
`ref`	Refreshment rate for BPS.
`rj_val`	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.
`ppi`	Double for the prior probability of inclusion (ppi) for each parameter.
`nmax`	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.

burn Optional number of iterations to use for burnin. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switchs velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.rr.data <- function(beta, n, Sig, noise, interc = TRUE) {
p <- length(beta)-(interc == TRUE)
dataX <- MASS::mvrnorm(n=n,mu=rep(0,p),Sigma=Sig)
if(interc) {dataX <- cbind(1, dataX)}
dataY <- rep(0, n)
dataY <- dataX %*% as.vector(beta)+rnorm(n, sd = sqrt(noise))
return(list(dataX = dataX, dataY = dataY))
}
p <- 3;
n<- 120
beta <- c(0.5,0.5, rep(0,p-1))
set.seed(1)
data <- generate.rr.data(beta,n,diag(1,p+1), noise = 2, interc = FALSE)
dataX <- data$dataX; dataY <- data$dataY
## Not run:
set.seed(1)
ppi_val <- 1/4
res <- bps_n_rr(maxTime = 1, dataX = dataX, datay = dataY,
                 prior_sigma2 = 1e2, x0 = rep(0,p+1), theta0 = rep(0,p+1),
                 rj_val = 0.6, ppi = ppi_val, nmax = 1e5, ref = 0.1, burn = -1)

plot_pdmp(res, coords = 1:3, inds = 1:1e3)

## End(Not run)
```

---

bps_s_logit *bps_s_logit*

Description

Applies the Reversible Jump BPS Sampler with Velocities drawn Uniformly on the p-Sphere to a Logistic regression with dirac spike and slab distribution, as detailed in Reversible Jump PDMP Samplers for Variable Selection, 2020. For included variables an independent Gaussian prior is assumed with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
bps_s_logit(
  maxTime,
  dataX,
  datay,
  prior_sigma2,
  x0,
  theta0,
  ref = 0.01,
  rj_val = 0.6,
  ppi = 0.5,
  nmax = 1000000L,
  burn = -1L
)
```

Arguments

`maxTime`	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
`dataX`	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
`datay`	Vector of n observations of a 0, 1-valued variable y.
`prior_sigma2`	Double for the prior variance for included variables.
`x0`	Initial position of the regression parameter
`theta0`	Initial velocity for the sampler. This should be chosen with unit velocities on each component (regardless of sign).
`ref`	Double for the refreshment rate of the BPS.
`rj_val`	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.
`ppi`	Double for the prior probability of inclusion (ppi) for each parameter.
`nmax`	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.
`burn`	Optional number of iterations to use for burnin. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switchs velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

## Not run:
bps_fit <- bps_s_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                       prior_sigma2 = 10, theta0 = rep(0, p),
                       x0 = rep(0, p), ref = 0.1, rj_val = 0.6,
                       ppi = ppi)

gibbs_fit <- gibbs_logit(maxTime = 1, dataX = data$dataX, datay =data$dataY,
                         prior_sigma2 = 10,beta = rep(0,p), gamma =rep(0,p),
                         ppi = ppi)

plot_pdmp(bps_fit, coords = 1:2, inds = 1:1e4,burn = .1, nsamples = 1e4,
          mcmc_samples = t(gibbs_fit$beta*gibbs_fit$gamma))

## End(Not run)
```

---

bps_s_rr *bps_s_rr*

Description

Applies the Reversible Jump BPS Sampler with Velocities drawn Uniformly on the p-Sphere to a Robust Regression problem with dirac spike and slab prior. Included variables are given an independent Gaussian prior with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
bps_s_rr(
  maxTime,
```

```
    dataX,
    datay,
    prior_sigma2,
    x0,
    theta0,
    ref = 0.1,
    rj_val = 0.5,
    ppi = 0.5,
    nmax = 1000000L,
    burn = -1L
)
```

Arguments

maxTime	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
dataX	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
datay	Vector of n observations of a continuous response variable y.
prior_sigma2	Double for the prior variance for included variables.
x0	Initial position of the regression parameter
theta0	Initial velocity for the sampler.
ref	Refreshment rate for BPS.
rj_val	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.
ppi	Double for the prior probability of inclusion (ppi) for each parameter.
nmax	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.
burn	Optional number of iterations to use for burn-in. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switches velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.rr.data <- function(beta, n, Sig, noise, interc = TRUE) {
p <- length(beta)-(interc == TRUE)
dataX <- MASS::mvrnorm(n=n,mu=rep(0,p),Sigma=Sig)
if(interc) {dataX <- cbind(1, dataX)}
dataY <- rep(0, n)
dataY <- dataX %*% as.vector(beta)+rnorm(n, sd = sqrt(noise))
return(list(dataX = dataX, dataY = dataY))
}
p <- 3;
n<- 120
beta <- c(0.5,0.5, rep(0,p-1))
set.seed(1)
data <- generate.rr.data(beta,n,diag(1,p+1), noise = 2, interc = FALSE)
dataX <- data$dataX; dataY <- data$dataY
## Not run:
set.seed(1)
ppi_val <- 1/4
res <- bps_s_rr(maxTime = 1, dataX = dataX, datay = dataY,
                 prior_sigma2 = 1e2, x0 = rep(0,p+1), theta0 = rep(0,p+1),
                 rj_val = 0.6, ppi = ppi_val, nmax = 1e5)

plot_pdmp(res, coords = 1:3, inds = 1:1e3)

## End(Not run)
```

`cond_mean` *Calculate the mean conditioned on being in a specific model*

Description

Calculate the mean conditioned on being in a specific model

Usage

```
cond_mean(times, positions, thetas, theta_c, burnin = 1)
```

Arguments

`times`	Vector of event times from the PDMP trajectory
`positions`	Matrix of positions from the PDMP trajectory, each column should correspond to a position
`thetas`	Matrix of PDMP velocities
`theta_c`	Vector indicating the model to condition on, 1s for active variables and zeros for inactive variables
`burnin`	Number of events to use as burnin

Value

Returns the mean conditioned on being in model theta_c estimated using the PDMP trajectories.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                       prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)
## Not run:
b <- cond_mean(zigzag_fit$times, zigzag_fit$positions, zigzag_fit$theta, theta_c = c(1,rep(0,p-1)))

## End(Not run)
```

`gen_sample` *Generate samples for PDMP trajectory*

Description

Get samples from PDMP trajectories taking a fixed time discretisation.

Usage

```
gen_sample(positions, times, nsample, theta = NULL, burn = 1)
```

Arguments

`positions`	Matrix of positions from the PDMP trajectory, each column should correspond to a position
`times`	Vector of event times from the PDMP trajectory
`nsample`	Number of desired samples from the PDMP trajectory
`theta`	Optional Matrix of velocities from the PDMP trajectory, each column should correspond to a velocity
`burn`	Index to start the discretisation from. Default is 1.

Value

Returns a list with the following objects:

`x`: Matrix of extracted samples of the position (x) taken using a fixed time discretisation of the PDMP

`theta`: Matrix of extracted samples of the velocity (theta) taken using a fixed time discretisation of the PDMP

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                       prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)
## Not run:
samples <- gen_sample(zigzag_fit$positions, zigzag_fit$times, 10^4)

plot(zigzag_fit$positions[1,],zigzag_fit$positions[2,], type = 'l', xlab = 'x1', ylab = 'x2')
points(samples$xx[1,], samples$xx[2,], col='red', pch=20)

## End(Not run)
```

---

gibbs_logit *gibbs_logit*

Description

Applies the Collapsed Gibbs Sampler to a Logistic regression with dirac spike and slab distribution, as detailed in Reversible Jump PDMP Samplers for Variable Selection, 2020. For included variables an independent Gaussian prior is assumed with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`. Code makes use of the package set-up for Polya-Gamma simulation available at `https://github.com/jgscott/helloPG`.

Usage

```
gibbs_logit(
  dataX,
  datay,
  beta,
  gamma,
  ppi = 0.5,
  nsamples = 100000L,
  maxTime = 1e+08,
  prior_sigma2 = 10
)
```

Arguments

dataX	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
datay	Vector of n observations of a 0, 1-valued variable y.
beta	Initial position of the regression parameter
gamma	Initial model for the sampler. Enteries should either be 1s or 0s.
ppi	Double for the prior probability of inclusion (ppi) for each parameter.
nsamples	Maximum number of samples. Default value is 10^5, lower values should be chosen for memory constraints if less samples are desired.
maxTime	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
prior_sigma2	Double for the prior variance for included variables. Default 10.

Value

Returns a list with the following objects:

`beta`: Matrix of regression parameter samples, columns are samples.

`gamma`: Matrix of model parameter samples columns are samples.

`times`: computation times at sampled events - Useful for plotting computational efficiency.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
```

```
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX,
                           datay = data$dataY, prior_sigma2 = 10,
                           theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)

gibbs_fit <- gibbs_logit(maxTime = 1, dataX = data$dataX, datay =data$dataY,
                         prior_sigma2 = 10,beta = rep(0,p), gamma =rep(0,p),
                         ppi = ppi)
## Not run:
plot_pdmp(zigzag_fit, coords = 1:2, inds = 1:1e3,burn = .1,
          nsamples = 1e4, mcmc_samples =t(gibbs_fit$beta*gibbs_fit$gamma))

## End(Not run)
```

---

marginal_mean *Calculate the marginal mean*

Description

Calculate the marginal mean

Usage

```
marginal_mean(times, positions, thetas, marginals = NULL, burnin = 1)
```

Arguments

times	Vector of event times from the PDMP trajectory
positions	Matrix of positions from the PDMP trajectory, each column should correspond to a position
thetas	Matrix of PDMP velocities
marginals	Vector of indices to calculate the marginal means.
burnin	Number of events to use as burnin

Value

Returns the posterior mean of the parameter estimated using the PDMP trajectories.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                       prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)
## Not run:
b <- marginal_mean(zigzag_fit$times, zigzag_fit$positions, zigzag_fit$theta, marginals=1:p)

## End(Not run)
```

---

models_visited — *Count the number of times a model is visited*

Description

Count the number of times a model is visited

Usage

```
models_visited(thetas)
```

Arguments

`thetas`	Vector of model indicies from the PDMP trajectory or samples from an MCMC sampler

Value

Returns a Matrix with rows corresponding to models and a final column corresponding to the number of times the model is visited

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                           prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p),
                           rj_val = 0.6, ppi = ppi)
## Not run:
models_visited(zigzag_fit$theta)

## End(Not run)
```

`model_probabilities` *Calculate posterior probabilities of inclusion based on PDMP trajectories*

Description

Calculate either marginal probabilities of inclusions or posterior probabilities of specific models.

Usage

```
model_probabilities(times, thetas, models = NULL, marginals = NULL, burnin = 1)
```

Arguments

`times`	Vector of event times from the PDMP trajectory
`thetas`	Matrix of velocities from the PDMP trajectory, each column should correspond to a velocities
`models`	Optional Matrix of indicies where rows correspond to models. Will return probabilities of each model `prob_mod`.
`marginals`	Optional Vector of indices to calculate the marginal probabilities of inclusion. Will return probabilities of inclusion for variable index `marginal_prob`.
`burnin`	Number of events to use as burnin

Value

Returns a list with the following objects:

`prob_mod`: Vector of posterior model probabilities based on the PDMP trajectories

`marginal_prob`: Vector of marginal probabilities for inclusion

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                       prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)
## Not run:
a <- models_visited(zigzag_fit$theta)

# Work out probability of top 10 most visited models and all marginal inclusion probabilities
# specific model probabilities become trivially small for large dimensions
b <- model_probabilities(zigzag_fit$times, zigzag_fit$thetas,
                         models = a[1:10,1:p], marginals=1:p)

## End(Not run)
```

`plot_pdmp` *Plot PDMP dynamics and samples for posterior distributions*

Description

Plot marginal densities and joint pairs plots for trajectories and samples of PDMP samplers and optionally MCMC samples for comparison. Care should be taken when interpreting marginal KDE estimates on the diagonal as the bandwidth of the KDE has an impact on how the Dirac spike is visualised.

Usage

```
plot_pdmp(
  pdmp_res,
  coords = 1:2,
  inds = 1:10^3,
  nsamples = 10^3,
  burn = 0.1,
  mcmc_samples = NULL,
  pch = 20,
  cols = NULL
)
```

Arguments

pdmp_res	List of positions, times and velocities returned from a PDMP sampler
coords	Vector of coordinates to plot the marginal and joint distributions
inds	Vector of indices of the PDMP trajectories to plot.
nsamples	Number of samples to generate and use for marginal density estimates of the PDMP methods
burn	Percentage of events to use as burn-in. Should be between 0 and 1.
mcmc_samples	Optional Matrix of samples from an MCMC method. Each row should be a sample.
pch	The graphics parameter for off diagonal plots. Default is 20.
cols	Colours to be used for plotting the PDMPs and MCMC samples (in order).

Value

Generates a plot of the marginal density on the diagonal and pairs plots of the trajectories

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p
```

```
zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                        prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                            ppi = ppi)
gibbs_fit <- gibbs_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                          prior_sigma2 = 10,beta = rep(0,p), gamma = rep(0,p),
                          ppi = ppi)
## Not run:
plot_pdmp(zigzag_fit, coords = 1:2, inds = 1:10^3,burn = .1,
          nsamples = 1e4, mcmc_samples = t(gibbs_fit$beta*gibbs_fit$gamma))

## End(Not run)
```

---

plot_pdmp_multiple *Plot multiple PDMP dynamics and MCMC samples for posterior distributions*

Description

Plots to compare PDMP samplers and optionally MCMC samples.

Usage

```
plot_pdmp_multiple(
  list_pdmp,
  coords = 1:2,
  inds = 1:10^3,
  nsamples = 10^3,
  burn = 0.1,
  mcmc_samples = NULL,
  pch = 20,
  cols = NULL
)
```

Arguments

list_pdmp	List of PDMP sampler trajectories to plot
coords	Vector of coordinates to plot the marginal and joint distributions
inds	Vector of indices of the PDMP trajectories to plot.
nsamples	Number of samples to generate and use for marginal density estimates of the PDMP methods
burn	Percentage of events to use as burn-in. Should be between 0 and 1, default 0.1.
mcmc_samples	Optional Matrix of samples from an MCMC method. Each row should be a sample.
pch	The graphics parameter for off diagonal plots. Default is 20.
cols	Colours to be used for plotting the PDMPs and MCMC samples (in order).

Value

Generates a plot of the marginal density on the diagonal and pairs plots of the trajectories

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX,datay =data$dataY,
                           prior_sigma2 = 10,theta0 = rep(0, p),
                           x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)
## Not run:
bps_fit <- bps_n_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                       prior_sigma2 = 10, theta0 = rep(0, p), x0 = rep(0, p),
                       ref = 0.1, rj_val = 0.6,ppi = ppi)

plot_pdmp_multiple(list(zz=zigzag_fit,bps=bps_fit), coords = 1:2, inds = 1:10^3,
                      nsamples = 1e4, burn = .1)

## End(Not run)
```

`zigzag_logit` *zigzag_logit*

Description

Applies the Reversible Jump ZigZag Sampler to a Logistic regression with dirac spike and slab distribution, as detailed in Reversible Jump PDMP Samplers for Variable Selection, 2020. For included variables an independent Gaussian prior is assumed with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
zigzag_logit(
  maxTime,
  dataX,
  datay,
  prior_sigma2,
  x0,
  theta0,
  rj_val = 0.6,
  ppi = 0.5,
  nmax = 1000000L,
  burn = -1L
)
```

Arguments

`maxTime`	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
`dataX`	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
`datay`	Vector of n observations of a 0, 1-valued variable y.
`prior_sigma2`	Double for the prior variance for included variables.
`x0`	Initial position of the regression parameter
`theta0`	Initial velocity for the sampler (Default has 1s on all components). This should be chosen with unit velocities on each component (regardless of sign).
`rj_val`	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.
`ppi`	Double for the prior probability of inclusion (ppi) for each parameter.
`nmax`	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.
`burn`	Optional number of iterations to use for burnin. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switchs velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                      prior_sigma2 = 10,theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                          ppi = ppi)

gibbs_fit <- gibbs_logit(maxTime = 1, dataX = data$dataX, datay = data$dataY,
                        prior_sigma2 = 10,beta = rep(0,p), gamma = rep(0,p),
                        ppi = ppi)
## Not run:
plot_pdmp(zigzag_fit, coords = 1:2, inds = 1:1e3,burn = .1, nsamples = 1e4,
           mcmc_samples = t(gibbs_fit$beta*gibbs_fit$gamma))

## End(Not run)
```

---

zigzag_logit_ss *zigzag_logit_ss*

Description

Applies the Reversible Jump ZigZag Sampler with subsampling to a Logistic regression with dirac spike and slab distribution, as detailed in Reversible Jump PDMP Samplers for Variable Selection, 2020. For included variables an independent Gaussian prior is assumed with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
zigzag_logit_ss(
  maxTime,
  dataX,
  datay,
```

```
  prior_sigma2,
  x0,
  theta0,
  cvref,
  rj_val = 0.6,
  ppi = 0.5,
  nmax = 1000000L,
  burn = -1L
)
```

Arguments

maxTime	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
dataX	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
datay	Vector of n observations of a 0, 1-valued variable y.
prior_sigma2	Double for the prior variance for included variables.
x0	Initial position of the regression parameter
theta0	Initial velocity for the sampler (Default has 1s on all components). This should be chosen with unit velocities on each component (regardless of sign).
cvref	Control variate vector of dimension p for subsampling. If no control variate set to a vector of zeros.
rj_val	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.
ppi	Double for the prior probability of inclusion (ppi) for each parameter.
nmax	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.
burn	Optional number of iterations to use for burnin. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switchs velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.logistic.data <- function(beta, n.obs, Sig) {
```

```
p <- length(beta)
dataX <- MASS::mvrnorm(n=n.obs,mu=rep(0,p),Sigma=Sig)
vals <- dataX %*% as.vector(beta)
generateY <- function(p) { rbinom(1, 1, p)}
dataY <- sapply(1/(1 + exp(-vals)), generateY)
return(list(dataX = dataX, dataY = dataY))
}

n <- 15
p <- 25
beta <- c(1, rep(0, p-1))
Siginv <- diag(1,p,p)
Siginv[1,2] <- Siginv[2,1] <- 0.9
set.seed(1)
data <- generate.logistic.data(beta, n, solve(Siginv))
ppi <- 2/p

## Not run:
zigzag_fit <- zigzag_logit(maxTime = 1, dataX = data$dataX,
                           datay = data$dataY, prior_sigma2 = 10,
                           theta0 = rep(0, p), x0 = rep(0, p), rj_val = 0.6,
                           ppi = ppi)

zigzag_fit_s <- zigzag_logit_ss(maxTime = 1, dataX = data$dataX,
                                datay = data$dataY,prior_sigma2 = 10,
                                theta0 = rep(0, p), x0 = rep(0, p),
                                rj_val = 0.6, cvref = c(1,rep(0,p-1)),
                                ppi = ppi)

gibbs_fit <- gibbs_logit(maxTime = 1, dataX = data$dataX, datay =data$dataY,
                         prior_sigma2 = 10,beta = rep(0,p), gamma =rep(0,p),
                         ppi = ppi)

plot_pdmp_multiple(list(zigzag_fit,zigzag_fit_s), coords = 1:2, burn = .1,
                   inds = 1:1e2, nsamples = 1e4,
                   mcmc_samples = t(gibbs_fit$beta*gibbs_fit$gamma))

## End(Not run)
```

---

zigzag_rr *zigzag_rr*

Description

Applies the Reversible Jump ZigZag Sampler to a Robust Regression problem with dirac spike and slab prior. Included variables are given an independent Gaussian prior with variance `prior_sigma2` and mean zero, variables are given prior probabilities of inclusion `ppi`.

Usage

```
zigzag_rr(
  maxTime,
  dataX,
  datay,
  prior_sigma2,
  x0,
  theta0,
  rj_val = 0.5,
  ppi = 0.5,
  nmax = 1000000L,
  burn = -1L
)
```

Arguments

`maxTime`	Maximum runtime (in Seconds) of the algorithm; will terminate the code after a given computation time or nmax iterations of the algorithm is reached.
`dataX`	Matrix of all covariates where the i-th row corresponds to all p covariates x_i,1, ..., x_i,p of the i-th observation.
`datay`	Vector of n observations of a continuous response variable y.
`prior_sigma2`	Double for the prior variance for included variables.
`x0`	Initial position of the regression parameter
`theta0`	Initial velocity for the sampler (Default has 1s on all components). This should be chosen with unit velocities on each component (regardless of sign).
`rj_val`	Reversible jump parameter for the PDMP method. This value is fixed over all models and is interpreted as the probability to jump to a reduced model when a parameter hits zero.
`ppi`	Double for the prior probability of inclusion (ppi) for each parameter.
`nmax`	Maximum number of iterations (simulated events) of the algorithm; will stop the algorithm when this number of iterations of the method have occured. Default value is 1e6, lower values should be chosen for memory constraints if less iterations are desired.
`burn`	Optional number of iterations to use for burnin. These are not stored so can be useful in memory intensive problems.

Value

Returns a list with the following objects:

`times`: Vector of event times where ZigZag process switchs velocity or jumps models.

`positions`: Matrix of positions at which event times occur, these are not samples from the PDMP.

`theta`: Matrix of new velocities at event times.

Examples

```
generate.rr.data <- function(beta, n, Sig, noise, interc = TRUE) {
p <- length(beta)-(interc == TRUE)
dataX <- MASS::mvrnorm(n=n,mu=rep(0,p),Sigma=Sig)
if(interc) {dataX <- cbind(1, dataX)}
dataY <- rep(0, n)
dataY <- dataX %*% as.vector(beta)+rnorm(n, sd = sqrt(noise))
return(list(dataX = dataX, dataY = dataY))
}
p <- 3;
n<- 120
beta <- c(0.5,0.5, rep(0,p-1))
set.seed(1)
data <- generate.rr.data(beta,n,diag(1,p+1), noise = 2, interc = FALSE)
dataX <- data$dataX; dataY <- data$dataY

set.seed(1)
ppi_val <- 1/4
res <- zigzag_rr(maxTime = 1, dataX = dataX, datay = dataY,
                 prior_sigma2 = 1e2, x0 = rep(0,p+1), theta0 = rep(0,p+1),
                 rj_val = 0.6, ppi = ppi_val, nmax = 1e5)
## Not run:
plot_pdmp(res, coords = 1:3, inds = 1:1e3)

## End(Not run)
```

Index