Package 'SDGLM'

January 20, 2026

Title Scalable Bayesian Inference for Dynamic Generalized Linear Models

Version 0.4.0

Description Implements scalable Markov chain Monte Carlo (Sca-MCMC) algorithms for Bayesian inference in dynamic generalized linear models (DGLMs). The package supports Pareto-type and Gamma-type DGLMs, which are suitable for modeling heavy-tailed phenomena such as wealth allocation and financial returns. It provides simulation tools for synthetic DGLM data, adaptive mutation-rate strategies (ScaI, ScaII, ScaIII), geometric temperature ladders for parallel tempering, and posterior predictive evaluation metrics (e.g., R2, RMSE). The methodology is based on the scalable MCMC framework described in Guo et al. (2025).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports stats, MASS, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Guangbao Guo [aut, cre],
X. Meggie Wen [aut],
Lixing Zhu [aut]

Maintainer Guangbao Guo <ggb11111111@163.com>

Repository CRAN

Date/Publication 2026-01-20 10:50:11 UTC

Contents

`compute_metrics` *Posterior-Predictive Metrics for Sca-MCMC Fit*

Description

Computes R2, RMSE, AIC and BIC after discarding the first 50 as burn-in. Predictive expectations are obtained by plugging the posterior mean coefficients into the appropriate inverse-link function.

Usage

```
compute_metrics(fit, y, X)
```

Arguments

`fit`	Object returned by ‘sca_mcmc‘ (must contain ‘beta_chain‘ and ‘family‘).
`y`	Observed response vector (length n).
`X`	n x p design matrix used for the fit.

Value

A data.frame with columns: R2, RMSE, AIC, BIC.

Examples

```
set.seed(123)
X <- matrix(rnorm(100 * 3), 100, 3)
beta <- c(0.5, -0.2, 0.1)
y <- rgamma(100, shape = 2, rate = exp(X %*% beta))
fit <- sca_mcmc(y, X, family = "gamma", method = "ScaII", iter = 1000)
vals <- compute_metrics(fit, y, X)
print(vals)
```

`compute_mutation_rate` *Compute Scalable Mutation-Rate Vector*

Description

Implements the three mutation-rate strategies (ScaI, ScaII, ScaIII) described in the reference paper.

Usage

```
compute_mutation_rate(
  method = c("ScaI", "ScaII", "ScaIII"),
  beta_star,
  beta_init,
  L,
  N_chain
)
```

Arguments

`method`	Character, one of "ScaI", "ScaII", "ScaIII".
`beta_star`	Target parameter vector (binary or factor).
`beta_init`	Current/initial parameter vector.
`L`	Integer > 0, length of the parameter vector.
`N_chain`	Integer > 1, number of parallel chains.

Value

List with components:

`Q`	Mutation-rate vector (length 2, r, or N_chain).
`Q0`	Base mutation rate (scalar).
`r`	Scalar, present only for ScaII - number of active components.
`w`	Vector, present only for ScaIII - normalized random weights.

Examples

```
beta_star <- c(1, 0, 1, 1, 0)
beta_init <- c(1, 1, 1, 0, 0)
compute_mutation_rate("ScaII", beta_star, beta_init, L = 5, N_chain = 8)
```

`dglm_likelihood` *Calculate Log-Likelihood for DGLM*

Description

Element-wise log-likelihood for Dynamic Generalized Linear Models with linear predictor eta = X * beta.

Usage

```
dglm_likelihood(
  y,
  X,
  beta,
  family = c("poisson", "pareto", "gamma"),
  alpha_gamma = 2,
  lambda_min = 1e-06
)
```

Arguments

`y`	Numeric response vector (length n).
`X`	Numeric design matrix (n x p).
`beta`	Numeric coefficient vector (length p).
`family`	Character distribution choice: "poisson", "pareto", or "gamma".
`alpha_gamma`	Shape parameter for Gamma (default 2).
`lambda_min`	Lower bound for Pareto shape lambda (default 1e-6).

Value

Numeric vector of length n; -Inf for illegal observations.

Examples

```
set.seed(123)
X <- matrix(rnorm(100 * 3), 100, 3)
beta <- c(0.5, -0.2, 0.1)

# Poisson
y_pois <- rpois(100, lambda = exp(X %*% beta))
ll_pois <- dglm_likelihood(y_pois, X, beta, family = "poisson")

# Gamma
y_gamma <- rgamma(100, shape = 3, rate = exp(X %*% beta))
ll_gamma <- dglm_likelihood(y_gamma, X, beta, family = "gamma", alpha_gamma = 3)
```

generate_temperature *Generate Geometric Inverse-Temperature Ladder Constructs a geometric sequence of temperatures (inverse temperatures) for parallel-tempering MCMC.*

Description

Generate Geometric Inverse-Temperature Ladder Constructs a geometric sequence of temperatures (inverse temperatures) for parallel-tempering MCMC.

Usage

```
generate_temperature(N_chain, T_max = 10)
```

Arguments

N_chain	Number of parallel chains (N_chain > 1).
T_max	Highest temperature (coldest chain). Default is 10.

Value

A numeric vector of length N_chain containing the geometrically spaced temperatures.

Examples

```
set.seed(1)
temps <- generate_temperature(N_chain = 8, T_max = 10)
print(temps)
```

geoTemp *Generate Geometric Temperature Ladder for Parallel Tempering*

Description

Produces a geometric progression of inverse-temperatures (or temperatures) commonly used in parallel-tempered MCMC algorithms.

Usage

```
geoTemp(N, T1 = 1, TN = 20)
```

Arguments

N	Integer > 1, number of chains/temperatures.
T1	Numeric > 0, coldest temperature (usually 1).
TN	Numeric > T1, hottest temperature.

Value

Numeric vector of length N containing the geometrically spaced temperatures T_1, T_2, ..., T_N.

Examples

```
geoTemp(8, T1 = 1, TN = 20)
```

`hamming_distance` *Normalized Hamming Distance*

Description

Computes the proportion of mismatches between two binary or factor vectors of equal length.

Usage

```
hamming_distance(beta_star, beta_init)
```

Arguments

`beta_star`	Target vector (true value).
`beta_init`	Initial/candidate vector.

Value

Scalar in [0,1] giving the fraction of differing positions.

Examples

```
hamming_distance(c(1, 0, 1, 0), c(1, 1, 1, 0))  # 0.25
```

`mutRate` *Scalable Mutation-Rate Strategies for Sca-MCMC*

Description

Computes the mutation-rate vector Q_P according to three scalable schemes proposed in the reference paper.

Usage

```
mutRate(type = c("ScaI", "ScaII", "ScaIII"), N, L, beta.star, beta0)
```

Arguments

type	Character, one of "ScaI", "ScaII", "ScaIII".
N	Integer > 1, number of parallel chains.
L	Integer > 0, length of the parameter vector.
beta.star	Target parameter vector (binary or factor).
beta0	Initial parameter vector (same length as beta.star).

Value

Numeric vector of length 2, r, or N depending on type, summing to Q0 and proportional to the chosen strategy.

Examples

```
beta.star <- c(1, 0, 1, 1, 0)
beta0 <- c(1, 1, 1, 0, 0)
mutRate("ScaII", N = 8, L = 5, beta.star, beta0)
```

---

print.SDGLM *Print method for SDGLM objects*

Description

Print method for SDGLM objects

Print method for SDGLM objects

Usage

```
## S3 method for class 'SDGLM'
print(x, ...)

## S3 method for class 'SDGLM'
print(x, ...)
```

Arguments

x	An SDGLM object
...	Additional arguments

Value

The input object ‘x‘ is returned invisibly. This function is called primarily for its side effect of printing a summary of the SDGLM object to the console.

print.summary.SDGLM *Print method for summary.SDGLM*

Description

Print method for summary.SDGLM

Print method for summary.SDGLM objects

Usage

```
## S3 method for class 'summary.SDGLM'
print(x, ...)

## S3 method for class 'summary.SDGLM'
print(x, ...)
```

Arguments

x	A summary.SDGLM object
...	Additional arguments

Value

The input object ‘x‘ is returned invisibly. This function is called primarily for its side effect of printing a formatted summary of the SDGLM model to the console.

rinvwishart *Generate Random Samples from the Inverse Wishart Distribution*

Description

Implements Bartlett decomposition to sample from the inverse Wishart distribution IW(v, S), where v = degrees of freedom and S = scale matrix.

Usage

```
rinvwishart(n = 1, v, S)
```

Arguments

n	Integer (>=1). Number of inverse Wishart samples to generate (default = 1).
v	Numeric (scalar). Degrees of freedom; must satisfy v > p - 1 (where p = ncol(S)).
S	Numeric matrix (p x p). Positive-definite scale matrix.

Value

If n=1: p x p inverse Wishart sample matrix. If n>1: 3D array (p x p x n) of independent inverse Wishart samples.

Examples

```
# 1 sample from IW(v=5, S=diag(2))
set.seed(123)
Sigma <- rinvwishart(n = 1, v = 5, S = diag(2))

# 10 samples (3D array)
Sigma_arr <- rinvwishart(n = 10, v = 5, S = diag(2))
```

---

sca_mcmc *Scalable MCMC for Dynamic GLMs*

Description

Implements Algorithm 1 of the reference paper with three mutation-rate strategies (ScaI/II/III) and three move types (mutation/crossover/exchange).

Usage

```
sca_mcmc(
  y,
  X,
  family = c("poisson", "pareto", "gamma"),
  method = c("ScaI", "ScaII", "ScaIII"),
  N_chain = 6L,
  iter = 10000L,
  burn = NULL,
  thin = 1L,
  T_max = 10,
  beta_init = NULL,
  verbose = TRUE
)
```

Arguments

y	Numeric response vector (length n).
X	n x p design matrix.
family	Character: "poisson", "pareto", or "gamma".
method	Character: "ScaI", "ScaII", or "ScaIII".
N_chain	Integer >= 2, number of parallel chains.
iter	Integer, total MCMC iterations per chain.

burn	Integer, burn-in length (default = iter/2).
thin	Integer, thinning interval (default = 1).
T_max	Numeric > 1, hottest temperature (default = 10).
beta_init	Optional matrix (N_chain x p) of initial coefficients. If NULL, random starts are generated.
verbose	Logical, print progress bar (default = TRUE).

Value

List with components:

beta_chain	3-D array (iter/thin x p x N_chain) of posterior samples.
family	Character, distribution family used.
method	Character, mutation-rate strategy used.

Examples

```
set.seed(1)
X <- matrix(rnorm(200 * 3), 200, 3)
beta <- c(0.5, -0.2, 0.1)
y <- rgamma(200, shape = 2, rate = exp(X %*% beta))
fit <- sca_mcmc(y, X, family = "gamma", method = "ScaII",
                N_chain = 6, iter = 1000, burn = 500)
```

---

sca_mcmc1 *Alternative Sca-MCMC Implementation for Variable Selection*

Description

An alternative implementation of Sca-MCMC for variable selection with binary inclusion indicators.

Usage

```
sca_mcmc1(
  y,
  X,
  family = c("poisson", "pareto", "gamma"),
  method = c("ScaI", "ScaII", "ScaIII"),
  N_chain = 8,
  n_iter = 5000,
  beta_star = NULL,
  alpha_gamma = 2
)
```

Arguments

y	Numeric response vector.
X	Design matrix (n x p).
family	Character, one of "poisson", "pareto", "gamma".
method	Mutation strategy: "ScaI", "ScaII", or "ScaIII".
N_chain	Number of parallel tempered chains (>1).
n_iter	Total MCMC iterations.
beta_star	Target (true) inclusion vector (for adaptive Q0; optional).
alpha_gamma	Shape parameter if family = "gamma" (default 2).

Value

List containing:

beta_chain	Array [n_iter x p x N_chain] of sampled inclusion vectors.
family	Model family used.
method	Mutation strategy used.

---

simGamma *Simulate Gamma Dynamic GLM*

Description

Generates a dynamic regression dataset for gamma family GLM where the rate parameter follows a dynamic process.

Usage

```
simGamma(N = 1000L, p = 4L, alpha_gamma = 2)
```

Arguments

N	Integer > 1, number of observations.
p	Integer > 0, number of predictors.
alpha_gamma	Numeric > 0, shape parameter for gamma distribution (default = 2).

Value

List with components:

X	N x p design matrix.
Y	Numeric vector of length N, gamma distributed observations.
beta	True coefficient vector.
alpha_gamma	Shape parameter used.

Examples

```
set.seed(3)
dat <- simGamma(N = 500, p = 3)
hist(dat$Y, main = "Gamma Data")
```

`simPareto` *Simulate Pareto-type Dynamic GLM*

Description

Generates a dynamic time-series where y_t = 1 + Gamma(shape = 1, scale = 1/lambda_t), and the inverse-scale lambda_t follows a stationary AR(1) process.

Usage

```
simPareto(N = 1000L, q = 4L)
```

Arguments

`N`	Integer > 1, series length.
`q`	Integer >= 1, number of predictors (used only for interface compatibility; no covariates are currently used in the generator).

Value

List with components:

`Y`	Numeric vector of length N, Pareto-type observations (y >= 1).
`lambda`	Numeric vector of length N, dynamic inverse-scale process.
`G`	AR(1) persistence coefficient (|G| < 1).
`sig2`	Innovation variance sigma^2.

Examples

```
set.seed(2)
dat <- simPareto(N = 500, q = 3)
plot(dat$lambda, type = "l")
```

`simPoisBin` *Simulate Poisson-Binomial Dynamic GLM*

Description

Generates a dynamic regression dataset where each response y_i | p_i ~ Binomial(n, p_i) with p_i = plogis(x_i' beta_i). The latent coefficients beta_i follow a Vector-AR(1) process.

Usage

```
simPoisBin(N = 1000L, n = 10L, q = 4L)
```

Arguments

`N`	Integer > 1, number of time points/individuals.
`n`	Integer > 0, binomial number of trials (constant across units).
`q`	Integer > 0, number of predictors (including intercept if desired).

Value

A list with components:

`X`	N x q design matrix.
`Y`	Integer vector of length N, counts of successes.
`beta`	N x q matrix of dynamic regression coefficients.
`G`	q x q autoregressive multiplier matrix (diagonal).
`Sig`	q x q innovation covariance matrix (diagonal).

Examples

```
set.seed(1)
dat <- simPoisBin(N = 500, n = 10, q = 4)
head(dat$Y)
```

`summary.SDGLM` *Summary method for SDGLM objects*

Description

Summary method for SDGLM objects

Summary method for SDGLM objects

Usage

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

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

Arguments

`object`	An SDGLM object
`...`	Additional arguments

Value

An object of class ‘summary.SDGLM‘, which is a list containing:

`coefficients`	A data frame with columns ‘Estimate‘ (posterior mean) and ‘Std.Error‘ (posterior standard deviation) for each parameter
`family`	Character string indicating the model family used
`method`	Character string indicating the mutation-rate strategy used

Index