Package: DSDRM (via r-universe)

July 10, 2026

Type Package

Title Distributed Sampling for Dynamic Regression Models

Version 0.1.3

Depends R (>= 4.1.0)

Description A toolbox for distributed dynamic regression modeling, parallel estimation, multiple distributed sampling algorithms (Metropolis-Hastings, block bootstrap, adaptive, hypergeometric), sparse matrix optimization, model visualization, prediction and performance evaluation. The philosophy of the package is described in Guo (2025) <doi:10.1038/s41598-025-93333-6>.

License Apache License 2.0

Encoding UTF-8

Imports parallel, Matrix

RoxygenNote 7.3.3

Author Guangbao Guo [aut, cre] (ORCID: <https://orcid.org/0000-0002-4115-6218>)

Maintainer Guangbao Guo <ggb11111111@163.com>

Config/testthat/edition 3

NeedsCompilation no

Language en-US

Suggests testthat

Repository https://cran.r-universe.dev

Date/Publication 2026-07-10 19:30:02 UTC

RemoteUrl https://github.com/cran/DSDRM

RemoteRef HEAD

RemoteSha 4d80006cb4a032033f99ce79958f2b7a024752ab

Contents

`adaptive_block_len` *Compute Adaptive Block Length*

Description

Compute adaptive optimal block length using autocorrelation and coefficient of variation. Designed for time-varying distributed regression models.

Usage

```
adaptive_block_len(Y_block, theta_block, L0 = 10)
```

Arguments

Y_block	Response variable in a single block.
theta_block	Parameter sequence in the block.
L0	Base block length.

Value

A list containing optimal block length, autocorrelation, and coefficient of variation.

Examples

```
Yb <- rnorm(50)
tb <- rnorm(50)
res <- adaptive_block_len(Yb, tb, L0 = 10)
res
```

---

block_data_split *Split Time Series Data into Distributed Blocks*

Description

Splits a full dataset into multiple contiguous blocks for distributed estimation.

Usage

```
block_data_split(Y, X, n_blocks)
```

Arguments

Y	Response vector
X	Design matrix
n_blocks	Number of blocks

Value

List of blocks containing Y and X

Examples

```
Y <- rnorm(100)
X <- matrix(rnorm(100*3), 100, 3)
blks <- block_data_split(Y, X, n_blocks = 4)
length(blks)
```

`calc_metrics` *Simulation Evaluation Metrics: MSE, MAE, R2, and Model Selection Accuracy*

Description

Computes key evaluation metrics for parameter estimation and model selection performance.

Usage

```
calc_metrics(theta_est, theta_true, Y_est, Y_true, gamma_est, gamma_true)
```

Arguments

`theta_est`	Estimated parameter vector.
`theta_true`	True parameter vector.
`Y_est`	Fitted response values.
`Y_true`	True response values.
`gamma_est`	Estimated model indicator vector.
`gamma_true`	True model indicator vector.

Value

Data frame containing MSE, MAE, R2, and model accuracy (ACC).

Examples

```
te <- c(0.8, 1.1, 0.05)
tt <- c(1, 1, 0)
ye <- rnorm(50)
yt <- rnorm(50)
ge <- c(1,1,0)
gt <- c(1,1,0)
calc_metrics(te, tt, ye, yt, ge, gt)
```

`dist_parallel_estimate` *Fully Distributed Parallel Estimation*

Description

High-performance parallel estimation for large-scale distributed data.

Usage

```
dist_parallel_estimate(blocks, gamma, cores = 2)
```

Arguments

blocks	List of data blocks
gamma	Model indicator
cores	CPU cores (default 2, limited to 2 for CRAN compatibility)

Value

List of estimates and runtime

Examples

```
sim <- dsdrm_generate_data(T_total = 200, K = 2, p = 5, W_beta = 0.2)
gam <- c(1,1,0,0,0)
res <- dist_parallel_estimate(sim$data_block, gamma = gam, cores = 2)
str(res)
```

---

dist_qn_algorithm *Distributed Quasi-Newton Iteration Main Algorithm*

Description

Implements the distributed quasi-Newton optimization algorithm for parameter estimation in time-varying distributed regression models.

Usage

```
dist_qn_algorithm(
  data_block,
  gamma,
  omega,
  lambda = 0.2,
  eps = 1e-06,
  max_iter = 1000,
  V = 0.5
)
```

Arguments

data_block	List of distributed data blocks.
gamma	Binary model indicator vector.
omega	Block weight vector.
lambda	L2 regularization parameter.
eps	Convergence tolerance.
max_iter	Maximum number of iterations.
V	Observation noise variance.

Value

List containing estimated parameters and iteration number.

Examples

```
sim <- dsdrm_generate_data(T_total = 300, K = 3, p = 4, W_beta = 0.1)
gam <- c(1,1,0,0)
w <- rep(1/3, 3)
fit <- dist_qn_algorithm(sim$data_block, gam, w, max_iter = 50)
fit$theta_est
```

---

dist_qn_with_time *DSDRM Algorithm with Computation Time Tracking*

Description

Runs the distributed quasi-Newton algorithm and records total execution time.

Usage

```
dist_qn_with_time(
  data_block,
  gamma,
  omega,
  lambda = 0.2,
  eps = 1e-06,
  max_iter = 1000
)
```

Arguments

data_block	List of distributed data blocks.
gamma	Binary model indicator vector.
omega	Block weight vector.
lambda	L2 regularization parameter.
eps	Convergence tolerance.
max_iter	Maximum number of iterations.

Value

List containing parameter estimates, iteration number, and computation time (seconds).

Examples

```
sim <- dsdrm_generate_data(T_total = 200, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
w <- c(0.4, 0.6)
res <- dist_qn_with_time(sim$data_block, gam, w, max_iter = 30)
res$time
```

---

dsdrm_fit *DSDRM Model Fitting (Main Interface)*

Description

Fits the Distributed Smooth Dynamic Regression Model (DSDRM).

Usage

```
dsdrm_fit(Y, X, n_blocks, gamma, lambda = 0.01)
```

Arguments

Y	Response vector
X	Design matrix
n_blocks	Number of distributed blocks
gamma	Model indicator vector
lambda	Regularization parameter

Value

Fitted DSDRM model object

Examples

```
Y <- rnorm(300)
X <- matrix(rnorm(300*4), 300, 4)
gam <- c(1,1,0,0)
mod <- dsdrm_fit(Y, X, n_blocks = 3, gamma = gam, lambda = 0.01)
mod$coefficients
```

`dsdrm_generate_data` *Generate Time-Varying Distributed Dynamic Regression Data*

Description

Generates simulated time series data for distributed dynamic regression models with time-varying coefficients, temporal drift, and distributed block structure.

Usage

```
dsdrm_generate_data(T_total, K, p, W_beta)
```

Arguments

`T_total`	Total length of the time series.
`K`	Number of data blocks for distributed computing.
`p`	Dimension of covariates.
`W_beta`	Variance of coefficient state noise.

Value

A list containing:

`data_block`	List of data blocks with Y, X, and block length Tk.
`T_total`	Total time length.
`K`	Number of blocks.
`p`	Covariate dimension.
`true_alpha`	True time-varying intercept sequence.
`true_beta`	True time-varying coefficient matrix.

Examples

```
sim_data <- dsdrm_generate_data(T_total = 200, K = 4, p = 5, W_beta = 0.2)
str(sim_data$data_block[[1]])
```

dsdrm_metrics *DSDRM Comprehensive Performance Metrics*

Description

Computes all key metrics for DSDRM estimation.

Usage

```
dsdrm_metrics(y_true, y_pred, theta_est, theta_true, gamma_est, gamma_true)
```

Arguments

y_true	True response
y_pred	Predicted response
theta_est	Estimated parameters
theta_true	True parameters
gamma_est	Estimated gamma
gamma_true	True gamma

Value

Data frame of metrics

Examples

```
yt <- rnorm(100)
yp <- rnorm(100)
te <- c(0.9, 1.05, 0.02)
tt <- c(1,1,0)
ge <- c(1,1,0)
gt <- c(1,1,0)
dsdrm_metrics(yt, yp, te, tt, ge, gt)
```

dsdrm_predict *Out-of-Sample Prediction for DSDRM*

Description

Generates predictions using fitted DSDRM model.

Usage

```
dsdrm_predict(model, X_new)
```

Arguments

model	Fitted dsdrm_fit object
X_new	New design matrix

Value

Predicted values

Examples

```
Y <- rnorm(200)
X <- matrix(rnorm(200*3), 200, 3)
gam <- c(1,1,0)
fit_mod <- dsdrm_fit(Y, X, n_blocks = 2, gamma = gam)
Xnew <- matrix(rnorm(50*3), 50, 3)
pred <- dsdrm_predict(fit_mod, Xnew)
head(pred)
```

---

dsdrm_sampling *DSDRM with Posterior Sampling (MCMC-based)*

Description

Generates parameter samples using Metropolis-Hastings within distributed framework.

Usage

```
dsdrm_sampling(blocks, gamma, omega, n_sample = 1000)
```

Arguments

blocks	List of data blocks
gamma	Model indicator
omega	Block weights
n_sample	Number of samples

Value

Matrix of parameter samples

Examples

```
sim <- dsdrm_generate_data(T_total = 150, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
w <- c(0.5, 0.5)
samp_mat <- dsdrm_sampling(sim$data_block, gam, w, n_sample = 100)
dim(samp_mat)
```

dynamic_coef_plot *Dynamic Coefficient Path Plot*

Description

Plots time-varying coefficient estimates from DSDRM.

Usage

```
dynamic_coef_plot(theta_seq, main = "DSDRM Dynamic Coefficients")
```

Arguments

theta_seq	Sequence of coefficient estimates
main	Title

Value

No return value, called for side effect (plotting dynamic coefficient path).

Examples

```
theta_path <- matrix(rnorm(200*3), 200, 3)
dynamic_coef_plot(theta_path, main = "Simulated Coefficients")
```

dynamic_opt_omega *Compute Dynamic Optimal Sampling Weights*

Description

Calculates optimal normalized weights for distributed blocks based on inverse MSE (mean squared error). Higher weights are assigned to blocks with lower estimation error.

Usage

```
dynamic_opt_omega(mse_vec)
```

Arguments

mse_vec	Vector of MSE values from each block.

Value

Normalized optimal weight vector summing to 1.

Examples

```
mse <- c(0.2, 0.5, 0.3)
w <- dynamic_opt_omega(mse)
sum(w)
```

`gamma_converge` *Convergence Check for Model Structure*

Description

Checks whether the model indicator gamma has converged by comparing the L1 distance between old and new gamma vectors.

Usage

```
gamma_converge(gamma_old, gamma_new, eps_gamma = 0.001)
```

Arguments

`gamma_old`	Previous model indicator vector.
`gamma_new`	Updated model indicator vector.
`eps_gamma`	Convergence threshold.

Value

Logical value (TRUE = converged, FALSE = not converged).

Examples

```
g1 <- c(1,1,0,0)
g2 <- c(1,1,0,0)
gamma_converge(g1, g2)
```

`gamma_fusion` *Fusion of Block-wise Model Selection Results (Weighted Voting)*

Description

Combines model selection results from all distributed blocks using weighted averaging. Constructs the final optimal model indicator gamma by weighted voting.

Usage

```
gamma_fusion(gamma_list, omega)
```

Arguments

gamma_list	List of binary model indicators from each block.
omega	Block weight vector.

Value

Fused optimal binary model indicator vector.

Examples

```
glist <- list(c(1,1,0), c(1,0,0))
w <- c(0.6, 0.4)
gamma_fusion(glist, w)
```

---

global_info_mat *Global Weighted Fisher Information Matrix*

Description

Computes the global weighted Fisher information matrix across all distributed blocks for active covariates selected by gamma.

Usage

```
global_info_mat(data_block, gamma, omega, V = 0.5)
```

Arguments

data_block	List of distributed data blocks.
gamma	Binary model indicator vector.
omega	Block weight vector.
V	Observation noise variance.

Value

Global information matrix of dimension p_eff x p_eff.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
w <- c(0.5, 0.5)
Gmat <- global_info_mat(sim$data_block, gam, w)
Gmat
```

global_mcmc_estimator *Global MCMC Estimator (Benchmark Method)*

Description

Implements a global Markov Chain Monte Carlo (MCMC) estimator used as a benchmark for comparison with the proposed DSDRM method.

Usage

```
global_mcmc_estimator(Y, X, iter = 1000)
```

Arguments

Y	Response vector.
X	Design matrix.
iter	Number of MCMC iterations.

Value

Combined vector of estimated intercept and coefficients.

Examples

```
Y <- rnorm(100)
X <- matrix(rnorm(100*3), 100, 3)
est_mcmc <- global_mcmc_estimator(Y, X, iter = 200)
est_mcmc
```

global_mcmc_with_time *Global MCMC Estimator with Computation Time Tracking*

Description

Runs the global MCMC benchmark and records execution time.

Usage

```
global_mcmc_with_time(Y, X, iter = 1000)
```

Arguments

Y	Response vector.
X	Design matrix.
iter	Number of MCMC iterations.

Value

List containing estimated parameters and computation time (seconds).

Examples

```
Y <- rnorm(100)
X <- matrix(rnorm(100*3), 100, 3)
out <- global_mcmc_with_time(Y, X, iter = 300)
out$time
```

`global_posterior_gamma`

Global Posterior Probability for Model Indicator

Description

Computes the global posterior probability of model indicator gamma using geometric averaging over all distributed blocks.

Usage

```
global_posterior_gamma(data_block, gamma, theta, omega, V = 0.5)
```

Arguments

`data_block`	List of distributed data blocks.
`gamma`	Binary model selection vector.
`theta`	Parameter vector (intercept + coefficients).
`omega`	Block weight vector.
`V`	Observation noise variance.

Value

Global posterior probability value.

Examples

```
sim <- dsdrm_generate_data(T_total = 120, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
th <- c(1, 1.2, 0.1)
w <- c(0.5,0.5)
pp <- global_posterior_gamma(sim$data_block, gam, th, w)
pp
```

global_quasi_ll *Global Weighted Quasi Log-Likelihood*

Description

Computes the global weighted quasi log-likelihood across all distributed blocks using block-specific optimal weights omega.

Usage

```
global_quasi_ll(data_block, theta, gamma, omega, V = 0.5)
```

Arguments

data_block	List of distributed data blocks.
theta	Parameter vector.
gamma	Binary model indicator vector.
omega	Block weight vector.
V	Observation noise variance.

Value

Global weighted quasi log-likelihood value.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
th <- c(1, 1.1, 0.05)
w <- c(0.5, 0.5)
ll <- global_quasi_ll(sim$data_block, th, gam, w)
ll
```

global_score *Global Score Vector*

Description

Computes the global score vector across all distributed blocks for time-varying distributed regression models.

Usage

```
global_score(data_block, theta, gamma, V = 0.5)
```

Arguments

data_block	List of distributed data blocks.
theta	Parameter vector (intercept + coefficients).
gamma	Binary model indicator vector.
V	Observation noise variance.

Value

Global score vector.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
th <- c(1, 1, 0)
sc <- global_score(sim$data_block, th, gam)
sc
```

---

init_gamma *Initialize Binary Model Indicator Vector Creates a binary (0/1) indicator vector for model selection. Sets 1 for active (true) covariates and 0 for irrelevant covariates.*

Description

Initialize Binary Model Indicator Vector Creates a binary (0/1) indicator vector for model selection. Sets 1 for active (true) covariates and 0 for irrelevant covariates.

Usage

```
init_gamma(p, active_idx)
```

Arguments

p	Total dimension of covariates.
active_idx	Indices of active (true) variables.

Value

A binary 0-1 vector of length p.

Examples

```
g <- init_gamma(p = 5, active_idx = c(1,2))
g
```

local_marginal_likelihood *Local Marginal Likelihood for One Block*

Description

Computes the local marginal likelihood for a single block under model selection indicator gamma.

Usage

```
local_marginal_likelihood(Y, X, gamma, theta, V = 0.5)
```

Arguments

Y	Response vector.
X	Design matrix.
gamma	Binary model indicator vector.
theta	Parameter vector (intercept + coefficients).
V	Observation noise variance.

Value

Scalar value of the marginal likelihood.

Examples

```
Y <- rnorm(50)
X <- matrix(rnorm(50*3),50,3)
gam <- c(1,1,0)
th <- c(1, 1.2, 0.1)
ml <- local_marginal_likelihood(Y, X, gam, th)
ml
```

local_quasi_ll *Local Quasi Log-Likelihood for One Block*

Description

Computes the local quasi log-likelihood for a single distributed block based on the selected model (gamma) and parameters (theta).

Usage

```
local_quasi_ll(Y, X, gamma, theta, V = 0.5)
```

Arguments

Y	Response vector.
X	Design matrix.
gamma	Binary model indicator vector.
theta	Parameter vector (intercept + active coefficients).
V	Observation noise variance.

Value

Scalar quasi log-likelihood value.

Examples

```
Y <- rnorm(50)
X <- matrix(rnorm(50*3), 50, 3)
gam <- c(1,1,0)
th <- c(1, 1.1, 0)
ll <- local_quasi_ll(Y, X, gam, th)
ll
```

---

mh_gamma_update *Dynamic Metropolis-Hastings Update for Model Structure*

Description

Performs a Metropolis-Hastings step to update the binary model indicator gamma. Used for dynamic model selection in distributed time-varying regression.

Usage

```
mh_gamma_update(gamma_curr, gamma_prop, data_block, theta, omega, V = 0.5)
```

Arguments

gamma_curr	Current model indicator vector.
gamma_prop	Proposed model indicator vector.
data_block	List of distributed data blocks.
theta	Parameter vector.
omega	Block weight vector.
V	Observation noise variance.

Value

List containing updated model indicator and acceptance rate.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
g_curr <- c(1,1,0)
g_prop <- c(1,0,0)
th <- c(1, 1, 0)
w <- c(0.5,0.5)
out <- mh_gamma_update(g_curr, g_prop, sim$data_block, th, w)
out$gamma
```

---

parallel_estimate *Parallel Block-wise Estimation Helper*

Description

Internal helper function for parallel distributed estimation. Computes closed-form OLS estimates for each block given selected model gamma.

Usage

```
parallel_estimate(block, gamma, V = 0.5)
```

Arguments

block	Single data block containing Y and X.
gamma	Binary model indicator vector.
V	Observation noise variance (not used in closed-form).

Value

Parameter estimates for the current block.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
blk <- sim$data_block[[1]]
gam <- c(1,1,0)
est <- parallel_estimate(blk, gam)
est
```

penalized_quasi_ll *L2 Penalized Quasi Log-Likelihood*

Description

Computes the L2 penalized global quasi log-likelihood for regularized estimation.

Usage

```
penalized_quasi_ll(data_block, theta, gamma, omega, lambda = 0.01, V = 0.5)
```

Arguments

data_block	List of distributed data blocks.
theta	Parameter vector.
gamma	Binary model indicator vector.
omega	Block weight vector.
lambda	L2 regularization parameter.
V	Observation noise variance.

Value

Penalized quasi log-likelihood value.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
gam <- c(1,1,0)
th <- c(1, 1, 0)
w <- c(0.5,0.5)
pll <- penalized_quasi_ll(sim$data_block, th, gam, w, lambda = 0.01)
pll
```

run_batch_simulation *Batch Simulation Main Function (Multivariate Grid Search)*

Description

Performs comprehensive batch simulations to compare estimation performance of DSDRM, Global MCMC, and Static Distributed methods under various settings.

Usage

```
run_batch_simulation(
  T_list = c(100, 500, 1000),
  K_list = c(2, 5, 10),
  W_list = c(0.1, 0.3, 0.5),
  repeat_num = 5,
  seed = NULL
)
```

Arguments

T_list	List of total time series lengths.
K_list	List of numbers of distributed blocks.
W_list	List of coefficient state noise variances.
repeat_num	Number of independent replications for each setting.
seed	Optional seed for reproducibility. If NULL (default), no seed is set.

Value

A data frame containing performance metrics (MSE, MAE, R2, ACC) for all methods under all simulation scenarios.

Examples

```
sim_out <- run_batch_simulation(T_list = c(150), K_list = c(2), W_list = c(0.1), repeat_num = 1)
head(sim_out)
```

---

run_batch_simulation_with_time

Batch Simulation with Runtime and Parallel Acceleration

Description

Conducts batch experiments for performance comparison among DSDRM, MCMC, Static Distributed, and Parallel DSDRM methods, with computation time recorded.

Usage

```
run_batch_simulation_with_time(
  T_list = c(1000, 5000, 10000),
  K_list = c(2, 5, 10),
  W_list = c(0.1, 0.3, 0.5),
  cores = 2,
  repeat_num = 3,
  seed = NULL
)
```

Arguments

T_list	Vector of time series lengths
K_list	Vector of number of distributed blocks
W_list	Vector of state noise variances
cores	Number of CPU cores for parallel computing
repeat_num	Number of independent replications
seed	Optional seed for reproducibility. If NULL (default), no seed is set.

Value

Data frame containing simulation settings, method name, runtime, and evaluation metrics (MSE, MAE, R2, ACC)

Examples

```
sim_res <- run_batch_simulation_with_time(
  T_list = c(200),
  K_list = c(2),
  W_list = c(0.1),
  repeat_num = 1
)
head(sim_res)
```

---

sparse_matrix_optim *Sparse Matrix Optimization for DSDRM*

Description

Accelerates information matrix and score computation using sparse matrices.

Usage

```
sparse_matrix_optim(blocks, gamma, omega)
```

Arguments

blocks	Data blocks
gamma	Model indicator
omega	Block weights

Value

Efficient information matrix

Examples

```
if(requireNamespace("Matrix", quietly = TRUE)){
  sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
  gam <- c(1,1,0)
  w <- c(0.5,0.5)
  Gsp <- sparse_matrix_optim(sim$data_block, gam, w)
  Gsp
}
```

`static_dist_with_time` *Static Distributed Estimator with Computation Time Tracking*

Description

Runs the static distributed benchmark estimator and records execution time.

Usage

```
static_dist_with_time(data_block)
```

Arguments

`data_block`	List of distributed data blocks.

Value

List containing estimated parameters and computation time (seconds).

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 2, p = 3, W_beta = 0.1)
out <- static_dist_with_time(sim$data_block)
out$time
```

`static_distributed_estimator`
Static Distributed Estimator (Benchmark Method)

Description

Implements a static distributed regression estimator as a benchmark. Estimates parameters locally on each block and averages results.

Usage

```
static_distributed_estimator(data_block)
```

Arguments

data_block	List of distributed data blocks.

Value

Averaged parameter vector across all blocks.

Examples

```
sim <- dsdrm_generate_data(T_total = 100, K = 3, p = 3, W_beta = 0.1)
est_static <- static_distributed_estimator(sim$data_block)
est_static
```

Index