# Package: factorDistADMM (via r-universe)

November 25, 2026

**Title** Distributed ADMM for High-Dimensional Static Factor Models

**Version** 0.1.6

**Description** Distributed Alternating Direction Method of Multipliers (ADMM) algorithms for high-dimensional static factor models. It provides a suite of distributed statistical estimators, including centralized Principal Component Analysis (PCA, C-PCA), distributed offline Principal Component Analysis (D-PCA), distributed online Expectation-Maximization (EM, D-EM), standard distributed Alternating Direction Method of Multipliers (S-ADMM), and the proposed penalized distributed online Alternating Direction Method of Multipliers (ADMM) with Lasso sparsity regularization. Trace-enabled versions of all algorithms are implemented to record the Frobenius estimation error and augmented Lagrangian values across iterations for convergence diagnosis. This package is suitable for large-scale high-dimensional data distributed over multiple computing nodes, where raw data cannot be gathered onto a single machine. It offers flexible simulation tools to generate static factor model data with sparse loadings and weak temporal dependence. The methodology of 'factorDistADMM' is described in Guo G. (2026) <doi:10.1016/j.jmva.2026.105688>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** MASS

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Depends** R (>= 3.5.0)

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

**Maintainer** Guangbao Guo <ggb11111111@163.com>

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

**Date/Publication** 2026-11-25 17:27:54 UTC

**RemoteUrl** https://github.com/cran/factorDistADMM

**RemoteRef** HEAD

**RemoteSha** baf6a45cb796661cd000ba76e124d4a636f57fc5

# Contents

C_PCA . . . . . 2
D_EM . . . . . 3
D_PCA . . . . . 4
frobenius_norm . . . . . 4
Ours_Dist_Online_ADMM . . . . . 5
Ours_Dist_Online_ADMM_trace . . . . . 5
S_ADMM . . . . . 6
S_ADMM_trace . . . . . 7
sim_static_factor . . . . . 8
soft_thresh . . . . . 9
split_distributed_data . . . . . 10

**Index** **11**

---

`C_PCA` *Centralized static principal component analysis*

---

## Description

This function performs centralized principal component analysis for a static high-dimensional factor model.

## Usage

```
C_PCA(X, r)
```

## Arguments

| | |
|---|---|
| `X` | Observation matrix with dimension $n \times T_{\text{sample}}$, where each column represents an observation. |
| `r` | Number of common factors. |

## Value

A list containing:

- `A_hat`: The estimated factor loading matrix of dimension $n \times r$.

## Examples

```
X <- matrix(stats::rnorm(100 * 10), nrow = 100, ncol = 10)
res_cpca <- C_PCA(X, r = 3)
```

---

D_EM *D_EM: Distributed online EM algorithm*

---

## Description

Distributed online EM for static factor model, following Gimenez-Febrer et al. 2020. No explicit Lasso penalty.

## Usage

```
D_EM(node_datasets, r, step_size = 0.01, max_iter = 100)
```

## Arguments

| | |
|---|---|
| node_datasets | A list. Each node contains submatrix, original row index. |
| r | Number of common factors. |
| step_size | Step size for online gradient update, default 0.01. |
| max_iter | Maximum number of iterations, default 100. |

## Value

A list containing:

- $A_hat$: $n \times r$ estimated factor loadings matrix.

## Examples

```
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
node_data <- split_distributed_data(X, K_n = 5)
res_dem <- D_EM(node_data, r = 3, step_size = 0.01, max_iter = 50)
```

## D_PCA *D_PCA: Distributed offline PCA*

### Description

Distributed offline PCA for factor model. Local covariance is computed at each node, then aggregated at the central server.

### Usage

```
D_PCA(node_datasets, r)
```

### Arguments

| | |
|---|---|
| node_datasets | A list. Each node contains submatrix, original row index. |
| r | Number of common factors. |

### Value

A list containing:

- $A_hat$: $n \times r$ estimated factor loadings matrix.

### Examples

```
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
node_data <- split_distributed_data(X, K_n = 5)
res_dpca <- D_PCA(node_data, r = 3)
```

## frobenius_norm *Frobenius norm of a matrix*

### Description

Compute the Frobenius norm of matrix $M$: $\|M\|_F = \sqrt{\sum_{i,j} M_{ij}^2}$

### Usage

```
frobenius_norm(M)
```

### Arguments

| | |
|---|---|
| M | A numeric matrix |

### Value

A scalar value of Frobenius norm

## Ours_Dist_Online_ADMM *Ours_Dist_Online_ADMM: Proposed distributed penalized-likelihood online ADMM*

### Description

The proposed method with Lasso penalty, soft-thresholding, stochastic approximation, simple outlier truncation and optimal weighted fusion.

### Usage

```
Ours_Dist_Online_ADMM(node_datasets, r, lambda, rho = 1, max_iter = 80)
```

### Arguments

| | |
|---|---|
| node_datasets | A list. Each node contains $X1k$(submatrix), $rowidx$(original row index). |
| r | Number of common factors. |
| lambda | Lasso sparsity tuning parameter, default 0.15. |
| rho | ADMM penalty parameter, default 1.0. |
| max_iter | Maximum number of iterations, default 80. |

### Value

A list containing:

- $A_hat$: $n \times r$ estimated factor loadings matrix.

### Examples

```
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
node_data <- split_distributed_data(X, K_n = 5)
res_ours <- Ours_Dist_Online_ADMM(node_data, r = 3, lambda = 0.15, rho = 1.0, max_iter = 50)
```

## Ours_Dist_Online_ADMM_trace *Ours_Dist_Online_ADMM_trace: Distributed penalized-likelihood online ADMM with convergence trace*

### Description

Extended version of the proposed distributed online ADMM. It includes Lasso penalty, soft-thresholding, Robbins-Monro stochastic approximation, simple outlier truncation and optimal weighted fusion. This variant records iteration-wise Frobenius estimation error and augmented Lagrangian value for convergence analysis.

## Usage

```
Ours_Dist_Online_ADMM_trace(
  node_datasets,
  A_true,
  r,
  lambda = 0.12,
  rho = 1,
  max_iter = 150
)
```

## Arguments

| | |
|---|---|
| node_datasets | A list. Each node contains $XIk$(submatrix), $rowidx$(original row index). |
| A_true | $n \times r$ true factor loading matrix used to compute estimation error. |
| r | Number of common factors. |
| lambda | Lasso sparsity tuning parameter, default 0.12. |
| rho | ADMM penalty parameter, default 1.0. |
| max_iter | Maximum number of iterations, default 150. |

## Value

A list containing:

- $A_hat$: $n \times r$ estimated factor loadings matrix.
- $trace$: A data frame storing iteration history, including iteration index, Frobenius estimation error and augmented Lagrangian.

## Examples

```
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
node_data <- split_distributed_data(X, K_n = 5)
A_true <- matrix(rnorm(100 * 3), nrow = 100, ncol = 3)
res_trace <- Ours_Dist_Online_ADMM_trace(
  node_data, A_true, r = 3, lambda = 0.12,
  rho = 1.0, max_iter = 50
)
```

---

S_ADMM *S_ADMM: Standard distributed ADMM*

---

## Description

Standard distributed ADMM minimizing negative log-likelihood. No Lasso sparsity penalty, no stochastic approximation, no outlier trimming.

### Usage

```
S_ADMM(node_datasets, r, rho = 1, max_iter = 80)
```

### Arguments

| | |
|---|---|
| node_datasets | A list. Each node contains submatrix, original row index. |
| r | Number of common factors. |
| rho | ADMM penalty parameter, default 1.0. |
| max_iter | Maximum number of iterations, default 80. |

### Value

A list containing:

- $A_hat$: $n \times r$ estimated factor loadings matrix.

### Examples

```
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
node_data <- split_distributed_data(X, K_n = 5)
res_sadmm <- S_ADMM(node_data, r = 3, rho = 1.0, max_iter = 50)
```

---

## S_ADMM_trace *S_ADMM_trace: Standard distributed ADMM with convergence trace*

---

### Description

Extended trace-recording version of the standard distributed ADMM baseline. It minimizes the negative log-likelihood without sparsity penalty, using local gradient iterations and simple averaging fusion across nodes. This function records Frobenius estimation error and augmented Lagrangian value at each iteration for convergence diagnosis.

### Usage

```
S_ADMM_trace(node_datasets, A_true, r, rho = 1, max_iter = 150)
```

### Arguments

| | |
|---|---|
| node_datasets | A list. Each node contains $XIk$(submatrix), $rowidx$(original row index). |
| A_true | $n \times r$ true factor loading matrix used to compute estimation error. |
| r | Number of common factors. |
| rho | ADMM penalty parameter, default 1.0. |
| max_iter | Maximum number of iterations, default 150. |

## Value

A list containing:

- $A_hat$: $n \times r$ estimated factor loadings matrix.
- $trace$: A data frame storing iteration history, including iteration index, Frobenius estimation error and augmented Lagrangian.

## Examples

```
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
node_data <- split_distributed_data(X, K_n = 5)
A_true <- matrix(rnorm(100 * 3), nrow = 100, ncol = 3)
res_trace <- S_ADMM_trace(node_data, A_true, r = 3, rho = 1.0, max_iter = 50)
```

---

# sim_static_factor *Simulate static high-dimensional factor model*

---

## Description

Generates data from static high-dimensional factor model: $X_t = AF_t + v_t$, where $v_t \sim N(0, V_t)$.

## Usage

```
sim_static_factor(n, r, T_sample, sparse_ratio, noise_decay)
```

## Arguments

| | |
|---|---|
| n | Number of observed dimensions |
| r | Number of common factors |
| T_sample | Time series sample size |
| sparse_ratio | Proportion of zero elements in the loadings matrix A; 0.3 in the paper |
| noise_decay | Geometric decay coefficient for the noise over time (0 < d < 1), inducing weak dependence; d = 0 corresponds to independent noise |

## Value

A list containing the simulated data and true model parameters:

| | |
|---|---|
| X | n by T_sample observed matrix; each column corresponds to $X_t$. |
| A_true | n by r true sparse factor loading matrix. |
| F_true | r by T_sample true latent factor matrix. |
| V_list | List of length T_sample, storing noise covariance matrix $V_t$ at each time point. |
| v | n by T_sample matrix of noise realisations. |

## Examples

```
# Generate simulated data from static high-dimensional factor model
sim_data <- sim_static_factor(n = 100,
                              r = 5,
                              T_sample = 2000,
                              sparse_ratio = 0.3,
                              noise_decay = 0.7)
# Extract observation matrix and true parameters
X <- sim_data$X
A_true <- sim_data$A_true
F_true <- sim_data$F_true
```

---

`soft_thresh` *Soft thresholding operator*

---

## Description

Element-wise soft-thresholding function widely used in the Alternating Direction Method of Multipliers (ADMM) for Lasso sparsity regularization.

## Usage

```
soft_thresh(x, lambda)
```

## Arguments

| | |
|---|---|
| `x` | numeric matrix or vector, input to be thresholded |
| `lambda` | Non-negative numeric scalar, thresholding parameter. |

## Value

Numeric matrix or vector with the same dimension and class as input `x`. Each element is calculated as `sign(x) * pmax(abs(x) - lambda, 0)`. Elements with absolute value less than `lambda` are shrunk to zero.

## Examples

```
x <- matrix(rnorm(10), nrow = 5)
soft_thresh(x, lambda = 0.2)
```

`split_distributed_data` *Split full data matrix into $K_n$ row blocks for distributed computing*

## Description

Split the high-dimensional observation matrix $X$ row-wise into $K_n$ submatrices, which can be used for distributed factor analysis.

## Usage

```
split_distributed_data(X, K_n = 5)
```

## Arguments

| | |
|---|---|
| `X` | Observation matrix with dimension $n \times T_{\text{sample}}$, each column is $X_t$. |
| `K_n` | Number of row partitions (computing nodes). Default is 5. |

## Value

A list of length $K_n$. Each element is a list containing:

- $XIk$: submatrix of $X$ on the corresponding row block;
- $rowidx$: original row indices of this block.

## Examples

```
# Generate test matrix
X <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
# Split into 5 row partitions
node_data <- split_distributed_data(X, K_n = 5)
# Check the number of partitions
length(node_data)
```

# Index

C_PCA, 2

D_EM, 3
D_PCA, 4

frobenius_norm, 4

Ours_Dist_Online_ADMM, 5
Ours_Dist_Online_ADMM_trace, 5

S_ADMM, 6
S_ADMM_trace, 7
sim_static_factor, 8
soft_thresh, 9
split_distributed_data, 10