

# Package ‘drrglm’

April 20, 2026

**Title** Doubly Regularized Matrix-Variate Regression

**Version** 0.3.2

**Maintainer** Zengchao Xu <zengc.xu@aliyun.com>

**Description** The doubly regularized matrix-variate regression solves a low-rank-plus-sparse structure for matrix-variate generalized linear models through a weighted combination of nuclear-norm and L1-norm.

The methodology implemented by this package is described in the paper “Doubly Regularized Matrix-Variate Regression”, which has been tentatively accepted for publication but does not yet have a DOI or URL. A formal citation will be added in a future update once the final publication details are available.

**Depends** R (>= 4.3.0)

**Imports** data.table, glmnet, Rcpp, stats

**LinkingTo** Rcpp, RcppArmadillo

**URL** <https://github.com/paradoxical-rhapsody/drrglm>

**BugReports** <https://github.com/paradoxical-rhapsody/drrglm/issues>

**License** AGPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**LazyData** true

**LazyDataCompression** xz

**NeedsCompilation** yes

**Author** Zengchao Xu [aut, cre, cph],  
Shan Luo [aut],  
Binyan Jiang [aut]

**Repository** CRAN

**Date/Publication** 2026-04-20 12:50:02 UTC

## Contents

drr . . . . .	2
EEG . . . . .	3
ini_paras . . . . .	4
simu_factor_model_paras . . . . .	5
simu_reg_coefs . . . . .	6
simu_zhouandli2014 . . . . .	7
tune.drrglm . . . . .	8
tune_drr_factor_model . . . . .	9
<b>Index</b>	<b>11</b>

---

drr *Doubly Regularized Matrix-Variate GLMs*

---

## Description

Solve the following problem

$$\min_{L,S} \left\{ \frac{1}{N} \sum_{n=1}^N \ell(y_n, X_n) + \lambda_1 \|L\|_* + \lambda_2 \|S\|_1 \right\},$$

where  $\ell(y_n, X_n)$  refers to the negative-log-likelihood on  $(y_n, X_n)$  under pre-specified GLM.

In linear model, the problem has the form

$$\min_{L,S} \left\{ \frac{1}{2N} \sum_{n=1}^N (y_n - \text{tr}(X_n' C))^2 + \lambda_1 \|L\|_* + \lambda_2 \|S\|_1 \right\}, \text{ s.t. } C = L + S.$$

## Usage

```
drrglm(
  x,
  y,
  family,
  lambda1,
  lambda2,
  C0 = NULL,
  tol = 0.001,
  maxIter = 300,
  verbose = FALSE
)
```

**Arguments**

<code>x</code>	$p_1 \times p_2 \times N$ numeric array with <b>mean zero</b> .
<code>y</code>	Numeric vector of length $N$ .
<code>family</code>	See <a href="#">glm</a> and <a href="#">family</a> .
<code>lambda1</code>	$\lambda_1$ .
<code>lambda2</code>	$\lambda_2$ .
<code>C0</code>	Initialization to $C$ .
<code>tol</code>	Convergence tolerance.
<code>maxIter</code>	Maximal step of iterations.
<code>verbose</code>	Print iterations?

**Value**

`list(L, S, Lsv, iter, lambda1, lambda2, isConvergent)`.

**Examples**

```
set.seed(2025)
r0 <- 13
c0 <- 17
N <- 500
x <- array(runif(r0*c0*N), c(r0, c0, N))
y <- rnorm(N)

family <- "gaussian"
lambda1 <- 0.15
lambda2 <- 0.04

system.time( egg <- drrglm(x, y, family, lambda1, lambda2) )
```

---

 EEG

*ElectroEncephaloGraphy Data*


---

**Description**

This data arises from a large study to examine EEG correlates of genetic predisposition to alcoholism. See details.

**Usage**

```
EEG
```

**Format**

```
list(alcoholic, control).
```

## Details

The original data are fully open-access and available in UCI Machine Learning Repository (<http://kdd.ics.uci.edu/databases/eeg/>). It includes two groups of subjects: 77 alcoholic and 45 control. In the original study, each subject was exposed to either a single stimulus (S1) or two stimuli (S1 and S2), which were pictures chosen from the 1980 Snodgrass and Vanderwart picture set. Each subject underwent 120 trials under each condition.

Here we provide this preprocessed data of the averages of 120 trials under S1 condition, which has been studied in several literature. The dataset is structured as a list containing two arrays,

- EEG\$alcoholic: Array of dimensions 256 x 64 x 77.
- EEG\$control: Array of dimensions 256 x 64 x 45.

## References

If you use this dataset, we would be very grateful if you could cite both the original data source and our work:

Zengchao Xu, Shan Luo, and Binyan Jiang. "Doubly Regularized Matrix-Variate Regression". *Submitted*.

## Examples

```
data(EEG, package="drrglm")
```

---

ini\_paras

*Initialize Parameters*

---

## Description

Initialize  $\hat{L}$  via nuclear-norm regularized regression.

## Usage

```
ini_paras(  
  x,  
  y,  
  family,  
  maxRank = min(floor(NROW(x)/2), floor(NCOL(x)/2)),  
  tol = 0.001,  
  verbose = FALSE  
)
```

**Arguments**

x	Numeric array.
y	Numeric vector.
family	See <a href="#">glm</a> and <a href="#">family</a> .
maxRank	The maximum of rank to be detected.
tol	Convergence tolerance.
verbose	Print iterations?

**Value**

list( family, grad, lipschitz, sigma0, rankStar, rankStarHat, lambda.star, L0.star )

---

simu\_factor\_model\_paras

*Simulation: Factor Model with Correlated Noise*

---

**Description**

$y = Bu + w$  with  $u \sim N(0, I_r)$ ,  $w \sim N(0, S)$  and  $B \in \mathbb{R}^{p \times r}$ .

- simu\_factor\_model\_paras: simulate (B, S).
- simu\_factor\_model\_data: simulate y.

**Usage**

simu\_factor\_model\_paras(p, r, noise, seed = 2023)

simu\_factor\_model\_data(N, B, S)

**Arguments**

p	Dimension of data.
r	Number of factors.
noise	Mode of $S$ : <ul style="list-style-type: none"> <li>• diag: Diagonal matrix.</li> <li>• rand: Nonzero elements at random positions.</li> <li>• tridiag: Tri-diagonal matrix.</li> <li>• block: Block matrix.</li> </ul>
seed	Random seed.
N	Sample size.
B	Matrix $p \times r$ .
S	Symmetric matrix $p \times p$ .

**Value**

- `simu_factor_model_paras`: list(B, S).
- `simu_factor_model_data`: Numeric matrix  $y$  of  $N \times p$ .

**Examples**

```
set.seed(2025)
N <- 50
p <- 15
r <- 7

paras <- simu_factor_model_paras(p, r, 'diag')
paras <- simu_factor_model_paras(p, r, 'rand')
paras <- simu_factor_model_paras(p, r, 'tridiag')
paras <- simu_factor_model_paras(p, r, 'block')

DT <- simu_factor_model_data(N, paras[["B"]], paras[["S"]])
```

---

simu\_reg\_coefs

*Simulation: Matrix-Variate GLM*


---

**Description**

Simulation: Matrix-Variate GLM

**Usage**

```
simu_reg_coefs(p1, p2, rank0, S.type, seed = 2023)

simu_reg_data(family, N, C0, err.student.dof = NULL)
```

**Arguments**

<code>p1</code>	Row dimension.
<code>p2</code>	Column dimension.
<code>rank0</code>	Rank of L.
<code>S.type</code>	Type of S. <ul style="list-style-type: none"> <li>• zero: Zero matrix.</li> <li>• rand: Nonzero elements at random positions.</li> <li>• block: Block matrix.</li> <li>• diag: Diagonal matrix.</li> </ul>
<code>seed</code>	Random seed.
<code>family</code>	See <a href="#">glm</a> and <a href="#">family</a> .
<code>N</code>	Sample size.

`C0` Coefficient matrix.  
`err.student.dof` The degree of freedom of Student-*t* for error term in family="gaussian". By default it is NULL for normal error.

### Value

- `simu_reg_coefs`: list(L, S).
- `simu_reg_data`: list(train=list(x, y), test=list(x, y)).

### Examples

```
p1 <- 10
p2 <- 9
rank0 <- 3

coefs <- simu_reg_coefs(p1, p2, rank0, "zero")
coefs <- simu_reg_coefs(p1, p2, rank0, "rand")
coefs <- simu_reg_coefs(p1, p2, rank0, "block")
coefs <- simu_reg_coefs(p1, p2, rank0, "diag")

N <- 100
S.type <- "rand"
family <- "binomial"

coefs <- simu_reg_coefs(p1, p2, rank0, S.type)
C0 <- coefs[["L"]] + coefs[["S"]]

set.seed(2025)
DT <- simu_reg_data(family, N, C0)
DT <- simu_reg_data("gaussian", N, C0, err.student.dof=3)
```

---

simu\_zhouandli2014      *Simulation: Setups in Regularized Matrix Regression*

---

### Description

Simulation: Setups in Regularized Matrix Regression

### Usage

```
simu_zhouandli2014(N, p1, p2, r0, s0, family)
```

**Arguments**

N	Sample size.
p1	Row dimension.
p2	Column dimension.
r0	Rank.
s0	signal proportion.
family	gaussian or binomial.

**Value**

```
list(C0, train=list(x, y), test=list(x, y)).
```

**References**

Zhou Hua and Li Lexin (2014). Regularized Matrix Regression. Journal of the Royal Statistical Society (Series B). [doi:10.1111/rssb.12031](https://doi.org/10.1111/rssb.12031)

**Examples**

```
N <- 100
p1 <- 64
p2 <- 64
r0 <- 1
s0 <- 0.05
family <- 'gaussian'

set.seed(2026)
DT <- simu_zhouandli2014(N, p1, p2, r0, s0, family)
```

---

tune.drrglm

*Doubly Regularized Regression for Matrix-Variate Data*

---

**Description**

Tune the  $(\lambda_1, \lambda_2)$  in `drr`.

**Usage**

```
tune_drrglm(
  iniParas,
  x,
  y,
  lambda1.factor = 1.3^seq(-3, 3, 0.2),
  maxCard = 30,
  tol = 0.001,
  maxIter = 500
)
```

**Arguments**

iniParas	Initial values returned by <code>ini_paras</code> . See examples below.
x	Numeric array.
y	Numeric vector.
lambda1.factor	Factor on initial guess $\lambda_1^2$ (By default $1.3^{\text{seq}(-3, 3, 0.2)}$ ).
maxCard	The maximal cardinality of $S$ .
tol	Convergence tolerance.
maxIter	Maximal step of iterations.

**Value**

List.

**Examples**

```
p1 <- 10
p2 <- 9
rank0 <- 3
S.type <- "rand"
coefs <- simu_reg_coefs(p1, p2, rank0, S.type)

N <- 500
family <- "gaussian"
L0 <- coefs[["L"]]
S0 <- coefs[["S"]]
C0 <- L0 + S0
DT <- simu_reg_data(family, N, C0)

x <- DT[["train"]][["x"]]
y <- DT[["train"]][["y"]]

lambda1.factor <- 1.1^(0:1)
maxRank <- 5

system.time( iniParas <- ini_paras(x, y, family, maxRank) )
system.time( drrObj <- tune_drrglm(iniParas, x, y, lambda1.factor) )
```

---

tune\_drr\_factor\_model *Simulating Data for Factor Model with Correlated Noise*

---

**Description**

Doubly regularized decomposition for covariance matrix in factor model.

**Usage**

```
tune_drr_factor_model(  
  y,  
  lambda1.factor = 1.1^(-2:2),  
  S.diag.penalize = FALSE,  
  tol = 0.001,  
  maxIter = 500  
)
```

**Arguments**

y	Numeric matrix of $N \times p$ .
lambda1.factor	Factor on $\lambda_1^8$ (by default $1.1^{(-2:2)}$ ).
S.diag.penalize	Whether to penalize the diagonal elements of S (by default FALSE).
tol	Convergence tolerance.
maxIter	Maximal step of iterations.

**Value**

List.

**Examples**

```
set.seed(2025)  
N <- 500  
p <- 30  
r <- 10  
noise <- "diag"  
paras <- simu_factor_model_paras(p, r, noise)  
y <- simu_factor_model_data(N, paras[["B"]], paras[["S"]])  
lambda1.factor <- 1.1^(-1:1)  
system.time( egg <- tune_drr_factor_model(y, lambda1.factor) )
```

# Index

## \* datasets

EEG, 3

drr, 2, 8

drrglm(drr), 2

EEG, 3

family, 3, 5, 6

glm, 3, 5, 6

ini\_paras, 4, 9

simu-factor-model

(simu\_factor\_model\_paras), 5

simu-reg-model(simu\_reg\_coefs), 6

simu\_factor\_model\_data

(simu\_factor\_model\_paras), 5

simu\_factor\_model\_paras, 5

simu\_reg\_coefs, 6

simu\_reg\_data(simu\_reg\_coefs), 6

simu\_zhouandli2014, 7

tune.drrglm, 8

tune\_drr\_factor\_model, 9

tune\_drrglm(tune.drrglm), 8