

# Package ‘netprioR’

October 16, 2025

**Title** A model for network-based prioritisation of genes

**Description** A model for semi-supervised prioritisation of genes integrating network data, phenotypes and additional prior knowledge about TP and TN gene labels from the literature or experts.

**Imports** stats, Matrix, dplyr, doParallel, foreach, parallel, sparseMVN, ggplot2, gridExtra, pROC

**Depends** methods, graphics, R(>= 3.3)

**Suggests** knitr, BiocStyle, pander

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, CellBasedAssays, Preprocessing, Network

**Type** Package

**Lazyload** yes

**LazyData** true

**Version** 1.34.0

**Date** 2016-05-08

**Author** Fabian Schmich

**Maintainer** Fabian Schmich <fabian.schmich@bsse.ethz.ch>

**License** GPL-3

**URL** <http://bioconductor.org/packages/netprioR>

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**git\_url** <https://git.bioconductor.org/packages/netprioR>

**git\_branch** RELEASE\_3\_21

**git\_last\_commit** 472209f

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.21

**Date/Publication** 2025-10-15

## Contents

netprioR-package . . . . .	2
bandwidth . . . . .	3
cmn . . . . .	3
conjugate_gradient . . . . .	4
cuthill_mckee . . . . .	4
fit . . . . .	5
laplacian . . . . .	6
learn . . . . .	6
netprioR-class . . . . .	7
norm_kern . . . . .	9
plot.netprioR . . . . .	9
ranks . . . . .	10
ROC . . . . .	11
simulate_labels . . . . .	11
simulate_network_random . . . . .	12
simulate_network_scalefree . . . . .	13
simulate_phenotype . . . . .	13
simulation . . . . .	14
weights . . . . .	15
<b>Index</b>	<b>16</b>

---

netprioR-package      *Package: netprioR*

---

### Description

This package provides a model for semi-supervised prioritisation of genes integrating network data, phenotypes and additional prior knowledge about TP and TN gene labels.

### Author(s)

Fabian Schmich | Computational Biology Group, ETH Zurich | <fabian.schmich@bsse.ethz.ch>

### References

Fabian Schmich et. al (2016).

---

bandwidth	<i>bandwidth</i>
-----------	------------------

---

**Description**

Compute the bandwidth of a matrix

**Usage**

```
bandwidth(x)
```

**Arguments**

x	Input matrix
---	--------------

**Value**

Bandwidth

**Author(s)**

Fabian Schmich

---

cmn	<i>Class Mass Normalization (CMN) from Zhu et al., 2003</i>
-----	---

---

**Description**

Class Mass Normalization (CMN) from Zhu et al., 2003

**Usage**

```
cmn(yhat, l, u)
```

**Arguments**

yhat	Response for labeled (l) and unlabeled (u) genes
l	Indices of labeled genes
u	Indices of unlabeled genes

**Value**

Class normalized yhat

**Author(s)**

Fabian Schmich

conjugate\_gradient      *Conjugate Gradient Solver*

---

**Description**

Solves linear equation systems iteratively

**Usage**

```
conjugate_gradient(A, b, x0 = rep(0, ncol(A)), threshold = 1e-15,  
  verbose = FALSE)
```

**Arguments**

A	Matrix
b	Coefficients
x0	Starting solution
threshold	Termination threshold
verbose	Show iterative progress

**Value**

Solution for equation system

**Author(s)**

Fabian Schmich

---

cuthill\_mckee      *Cuthill McKee (CM) algorithm*

---

**Description**

Transform sparse matrix into a band matrix

**Usage**

```
cuthill_mckee(x)
```

**Arguments**

x	Input matrix
---	--------------

**Value**

Band matrix

**Author(s)**

Fabian Schmich

---

`fit`*Fit netprioR model*

---

**Description**Fit `netprioR` model**Usage**

```
fit(object, ...)  
  
## S4 method for signature 'netprioR'  
fit(object, refit = FALSE, ...)
```

**Arguments**

<code>object</code>	A <code>netprioR</code> object
<code>...</code>	Additional arguments
<code>refit</code>	Flag whether to overwrite existing fit

**Value**A `netprioR` object with fitted model**Author(s)**

Fabian Schmich

**Examples**

```
data(simulation)  
np <- netprioR(networks = simulation$networks,  
               phenotypes = simulation$phenotypes,  
               labels = simulation$labels.obs,  
               model.fit = FALSE)  
  
summary(np)  
np <- fit(np, nrestarts = 1, verbose = FALSE)  
summary(np)
```

---

laplacian	<i>Graph Laplacian</i>
-----------	------------------------

---

**Description**

Compute the Laplacian matrix of a graph given its adjacency matrix

**Usage**

```
laplacian(x, norm = c("none", "sym", "asym"))
```

**Arguments**

x	Adjacency matrix
norm	Type of normalisation

**Value**

Laplacian matrix

**Author(s)**

Fabian Schmich

---

learn	<i>Fit netprioR model</i>
-------	---------------------------

---

**Description**

Infer parameters and hidden data using the EM algorithm of netprioR

**Usage**

```
learn(Yobs, X, G, l, u, a = 0.1, b = 0.1, sigma2 = 1, tau2 = 10,  
      eps = 1e-11, max.iter = 500, thresh = 0.001, use.cg = TRUE,  
      thresh.cg = 1e-05, nrestarts = 5, max.cores = detectCores(),  
      verbose = FALSE)
```

**Arguments**

Yobs	Observed labels (NA, if not observed)
X	Phenotypes
G	Graph Laplacians
l	Indices of labelled instances
u	Indices of unlabelled instances
a	Shape parameter of Gamma prior for W
b	Scale parameter of Gamma prior for W
sigma2	Cariance for Gaussian labels
tau2	Variance for Gaussian prior for beta
eps	Small value added to diagonal of Q in order to make it non-singular
max.iter	Maximum number of iterations for EM
thresh	Threshold for termination of EM with respect to change in parameters
use.cg	Flag whether to use conjugate gradient instead of exact computation of expectations
thresh.cg	Threshold for the termination of the conjugate gradient solver
nrestarts	Number of restarts for EM
max.cores	Maximum number of cores to use for parallel computation
verbose	Print verbose output

**Value**

List containing: Predicted labels  $\hat{Y}$  and inferred parameters  $W$  and  $\beta$

**Author(s)**

Fabian Schmich

---

netprioR-class      *netprioR*

---

**Description**

Class that represents a netprioR model.

**Usage**

```
netprioR(networks, phenotypes, labels, ...)

## S4 method for signature 'list,matrix,factor'
netprioR(networks, phenotypes, labels,
  fit.model = FALSE, a = 0.1, b = 0.1, sigma2 = 0.1, tau2 = 100,
  eps = 1e-10, max.iter = 500, thresh = 1e-06, use.cg = FALSE,
  thresh.cg = 1e-06, nrestarts = 5, max.cores = detectCores(),
  verbose = TRUE, ...)
```

**Arguments**

networks	List of NxN adjacency matrices of gene-gene similarities
phenotypes	Matrix of dimension NxP containing covariates
labels	Vector of Nx1 labels for all genes (NA if no label available)
...	Additional arguments
fit.model	Indicator whether to fit the model
a	Shape parameter of Gamma prior for W
b	Scale parameter of Gamma prior for W
sigma2	Cariance for Gaussian labels
tau2	Variance for Gaussian prior for beta
eps	Small value added to diagonal of Q in order to make it non-singular
max.iter	Maximum number of iterations for EM
thresh	Threshold for termination of EM with respect to change in parameters
use.cg	Flag whether to use conjugate gradient instead of exact computation of expectations
thresh.cg	Threshold for the termination of the conjugate gradient solver
nrestarts	Number of restarts for EM
max.cores	Maximum number of cores to use for parallel computation
verbose	Print verbose output

**Value**

A [netprioR](#) object

**Slots**

networks List of NxN adjacency matrices of gene-gene similarities  
 phenotypes Matrix of dimension NxP containing covariates  
 labels Vector of Nx1 labels for all genes. NA if no label available.  
 is.fitted Flag indicating if model is fitted  
 model List containing estimated parameters and imputed missing data

**Author(s)**

Fabian Schmich

**Examples**

```
# runs long-ish
data(simulation)
np <- netprioR(networks = simulation$networks,
               phenotypes = simulation$phenotypes,
               labels = simulation$labels.obs,
               fit.model = TRUE)
summary(np)
```

---

norm_kern	<i>Normalise kernel</i>
-----------	-------------------------

---

**Description**

adopted from GeneMania, Mostafavi et al, 2009

**Usage**

```
norm_kern(x)
```

**Arguments**

x	kernel
---	--------

**Value**

Normalised kernel

**Author(s)**

Fabian Schmich

---

plot.netprioR	<i>Plot method for netprioR objects</i>
---------------	---

---

**Description**

Plot method for [netprioR](#) objects

**Usage**

```
## S3 method for class 'netprioR'  
plot(x, which = c("all", "weights", "lik", "scores"), ...)
```

**Arguments**

x	A <a href="#">netprioR</a> object
which	Flag for which plot should be shown, options: weights, lik, scores, all
...	Additional paramters for plot

**Value**

Plot of the weights, likelihood, ranks, or all three

**Author(s)**

Fabian Schmich

**Examples**

```
data(simulation)
plot(simulation$model)
```

---

ranks

*Retrieve ranked prioritisation list*

---

**Description**

Retrieve ranked prioritisation list

**Usage**

```
ranks(object)

## S4 method for signature 'netprioR'
ranks(object)
```

**Arguments**

object            A [netprioR](#) object

**Value**

Ranked list of prioritised genes

**Author(s)**

Fabian Schmich

**Examples**

```
data(simulation)
ranks(simulation$model)
```

---

ROC	<i>Compute ROC curve from netprioR model and true labels</i>
-----	--

---

**Description**

Compute ROC curve from netprioR model and true labels

**Usage**

```
ROC(object, ...)  
  
## S4 method for signature 'netprioR'  
ROC(object, true.labels, plot = FALSE, ...)
```

**Arguments**

object	A <a href="#">netprioR</a> object
...	Additional arguments
true.labels	True full set of underlying labels
plot	Flag whether to plot the AUC curve

**Value**

ROC curve with AUC

**Author(s)**

Fabian Schmich

**Examples**

```
data(simulation)  
ROC(simulation$model, true.labels = simulation$labels.true)
```

---

simulate_labels	<i>Simulate labels</i>
-----------------	------------------------

---

**Description**

Simulate labels

**Usage**

```
simulate_labels(values, sizes, nobs)
```

**Arguments**

values	Vector of labels for groups
sizes	Vector of group sizes
nobs	Vector of number of observed labels per group

**Value**

List of Y, Yobs and indices for labeled instances

**Author(s)**

Fabian Schmich

**Examples**

```
labels <- simulate_labels(values = c("Positive", "Negative"),
  sizes = c(10, 10),
  nobs = c(5, 5))
```

---

simulate\_network\_random

*Simulate random networks with predefined number of members for each of the two groups and the number of neighbours for each node*

---

**Description**

Simulate random networks with predefined number of members for each of the two groups and the number of neighbours for each node

**Usage**

```
simulate_network_random(nmemb, nnei = 1)
```

**Arguments**

nmemb	Vector of number of members for each group
nnei	Number of neighbours for each node

**Value**

Adjacency matrix of graph

**Author(s)**

Fabian Schmich

**Examples**

```
network <- simulate_network_random(nmemb = c(10, 10), nnei = 1)
```

---

simulate\_network\_scalefree  
*Simulate scalefree networks*

---

**Description**

Simulate scale free networks for predefined number of members for each of two groups and a parameter pclus that determines how strictly distinct the groups are

**Usage**

```
simulate_network_scalefree(nmemb, pclus = 1)
```

**Arguments**

nmemb            Vector of numbers of members per group  
pclus            Scalar in [0, 1] determining how strictly distinct groups are

**Value**

Adjacency matrix

**Author(s)**

Fabian Schmich

**Examples**

```
network <- simulate_network_scalefree(nmemb = c(10, 10), pclus = 0.8)
```

---

simulate\_phenotype    *Simulate phenotypes correlated to labels pivoted into two groups*

---

**Description**

Simulate phenotypes correlated to labels pivoted into two groups

**Usage**

```
simulate_phenotype(labels.true, meandiff, sd)
```

**Arguments**

labels.true      Vector of labels  
meandiff        difference of means between positive and negative groups  
sd               Standard deviation of the phenotype

**Value**

Simulated phenotype

**Author(s)**

Fabian Schmich

**Examples**

```
data(simulation)
phenotypes <- simulate_phenotype(labels.true = simulation$labels.true, meandiff = 0.5, sd = 1)
```

---

simulation	<i>Example data: Simulated networks, phenotypes and labels for N = 1000 genes</i>
------------	---

---

**Description**

The data set contains simulated data for  $N = 1000$  genes and  $P = 1$  (univariate) phenotypes. The list of networks contains 2 low noise networks and two high noise networks. The class labels are "Positive" and "Negative".

**Usage**

```
data(simulation)
```

**Details**

The code used to simulate the data can be found in `system.file("example", "data_simulation.R", package = "netprioR")`

**Value**

List of simulated networks, phenotypes and labels for 1000 genes

---

weights	<i>Retrieve network weights</i>
---------	---------------------------------

---

**Description**

Retrieve network weights

**Usage**

```
weights(object, ...)
```

```
## S4 method for signature 'netprioR'  
weights(object)
```

**Arguments**

object	A <a href="#">netprioR</a> object
...	Additional arguments

**Value**

Estimated network weights

**Author(s)**

Fabian Schmich

**Examples**

```
data(simulation)  
weights(simulation$model)
```

# Index

- \* **package**
  - netprioR-package, 2
- bandwidth, 3
- cmn, 3
- conjugate\_gradient, 4
- cuthill\_mckee, 4
- fit, 5
- fit, netprioR-method (fit), 5
- laplacian, 6
- learn, 6
- netprioR, 5, 8-11, 15
- netprioR (netprioR-class), 7
- netprioR, list, matrix, factor-method (netprioR-class), 7
- netprioR-class, 7
- netprioR-package, 2
- norm\_kern, 9
- plot.netprioR, 9
- ranks, 10
- ranks, netprioR-method (ranks), 10
- ROC, 11
- ROC, netprioR-method (ROC), 11
- simulate\_labels, 11
- simulate\_network\_random, 12
- simulate\_network\_scalefree, 13
- simulate\_phenotype, 13
- simulation, 14
- weights, 15
- weights, netprioR-method (weights), 15