

# Package ‘greed’

March 18, 2022

**Type** Package

**Title** Clustering and Model Selection with the Integrated  
Classification Likelihood

**Version** 0.6.0

**Date** 2022-03-18

**URL** <https://comeetie.github.io/greed/>,  
<https://github.com/comeetie/greed>

**BugReports** <https://github.com/comeetie/greed/issues>

**Maintainer** Etienne Côme <etienne.come@univ-eiffel.fr>

**Description** An ensemble of algorithms that enable the clustering of networks and data matrices (such as counts, categorical or continuous) with different type of generative models. Model selection and clustering is performed in combination by optimizing the Integrated Classification Likelihood (which is equivalent to minimizing the description length). Several models are available such as: Stochastic Block Model, degree corrected Stochastic Block Model, Mixtures of Multinomial, Latent Block Model. The optimization is performed thanks to a combination of greedy local search and a genetic algorithm (see <arXiv:2002:11577> for more details).

**License** GPL

**Depends** R (>= 3.5.0)

**Imports** Rcpp (>= 1.0.0), Matrix, future, listenv, ggplot2, graphics,  
methods, stats, RSpecra, grid, gtable, gridExtra, cba, cli

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** testthat, MASS, knitr, rmarkdown, spelling, igraph,  
tidygraph, ggraph

**Encoding** UTF-8

**VignetteBuilder** knitr

**RoxygenNote** 7.1.2.9000

**Collate** 'RcppExports.R' 'tools\_misc.R' 'models\_classes.R'  
'fit\_classes.R' 'tools\_cleanpath.R' 'alg\_genetic.R'  
'alg\_hybrid.R' 'alg\_classes.R' 'alg\_multistart.R' 'data.R'  
'greed.R' 'model\_combinedmodels.R' 'model\_dclbm.R'

'model\_dcsbm.R' 'model\_diaggmm.R' 'model\_gmm.R' 'model\_lca.R'  
 'model\_mom.R' 'model\_mor.R' 'model\_multsbm.R' 'model\_sbm.R'  
 'tools\_generator.R' 'tools\_ploting.R'

**Language** en-US

**LazyData** true

**NeedsCompilation** yes

**Author** Etienne Côme [aut, cre],  
 Nicolas Jouvin [aut]

**Repository** CRAN

**Date/Publication** 2022-03-18 12:50:02 UTC

## R topics documented:

Alg-class	4
available_algorithms	4
available_models	4
Books	5
clustering	5
coef,DcLbmFit-method	6
coef,DcSbmFit-method	7
coef,DiagGmmFit-method	8
coef,GmmFit-method	8
coef,IclFit-method	9
coef,LcaFit-method	9
coef,MoMFit-method	10
coef,MoRFit-method	10
coef,MultSbmFit-method	11
coef,SbmFit-method	11
CombinedModels	12
CombinedModelsFit-class	13
CombinedModelsPath-class	13
cut,DcLbmPath-method	14
cut,IclPath-method	15
DcLbm	15
DcLbmFit-class	16
DcLbmPath-class	17
DcSbm	18
DcSbmFit-class	20
DcSbmPath-class	20
DiagGmm	22
DiagGmmFit-class	23
DiagGmmPath-class	24
DlvmCoPrior-class	25
DlvmPrior-class	25
extractSubModel	25
fashion	26

Fifa	27
Football	27
Genetic-class	28
Gmm	29
GmmFit-class	30
gmmpairs	31
GmmPath-class	31
greed	32
H	33
Hybrid-class	34
ICL	35
IclFit-class	35
IclPath-class	36
Jazz	36
K	37
Lca	37
LcaFit-class	38
LcaPath-class	39
MI	40
MoM	40
MoMFit-class	41
MoMPath-class	42
MoR	43
MoRFit-class	44
MoRPath-class	45
Multistarts-class	46
MultSbm	46
MultSbmFit-class	47
MultSbmPath-class	48
mushroom	49
Ndrangheta	49
NewGuinea	50
NMI	51
plot,DcLbmFit,missing-method	51
plot,DcLbmPath,missing-method	52
plot,DcSbmFit,missing-method	53
plot,DiagGmmFit,missing-method	53
plot,GmmFit,missing-method	54
plot,IclPath,missing-method	55
plot,LcaFit,missing-method	55
plot,MoMFit,missing-method	56
plot,MultSbmFit,missing-method	56
plot,SbmFit,missing-method	57
prior	58
rdcsbm	58
rlbm	59
rlca	60
rmm	61

rmreg . . . . .	62
rmultsbm . . . . .	63
rsbm . . . . .	64
Sbm . . . . .	65
SbmFit-class . . . . .	66
SbmPath-class . . . . .	67
Seed-class . . . . .	68
SevenGraders . . . . .	68
show,IclFit-method . . . . .	69
spectral . . . . .	69
to_multinomial . . . . .	70
Youngpeoplesurvey . . . . .	70

<b>Index</b>	<b>72</b>
--------------	-----------

---

Alg-class	<i>Abstract optimization algorithm class</i>
-----------	--

---

### Description

An S4 class to represent an abstract optimization algorithm.

---

available_algorithms	<i>Display the list of every currently available optimization algorithm</i>
----------------------	---

---

### Description

Display the list of every currently available optimization algorithm

### Usage

```
available_algorithms()
```

---

available_models	<i>Display the list of every currently available DLVM</i>
------------------	---

---

### Description

Display the list of every currently available DLVM

### Usage

```
available_models()
```

---

Books

*Books about US politics network dataset*

---

### Description

A network of books about US politics published around the time of the 2004 presidential election and sold by the online bookseller Amazon.com. Edges between books represent frequent co-purchasing of books by the same buyers. The network was compiled by V. Krebs and is unpublished, but can found on Krebs' web site. Thanks to Valdis Krebs for permission to post these data on this web site.

### Usage

```
data(Books)
```

### Format

An object of class `list` with two fields;

**X** network adjacency matrix as a `sparseMatrix` of size 105x105

**label** a factor of length (size 105) with levels "l", "n", or "c" to indicate whether the books are liberal, neutral, or conservative

### Source

[M. E. J. Newman Network datasets](#)

### Examples

```
data(Books)
```

---

clustering

*Method to extract the clustering results from an `IclFit-class` object*

---

### Description

This method take a `IclFit-class` object and return an integer vector with the cluster assignments that were found.

### Usage

```
clustering(fit)
```

```
## S4 method for signature 'IclFit'  
clustering(fit)
```

**Arguments**

fit                    an IclFit solution

**Value**

an integer vector with cluster assignments. Zero indicates noise points.

**Methods (by class)**

- IclFit: IclFit-class method

---

coef, DcLbmFit-method    *Extract parameters from an DcLbmFit-class object*

---

**Description**

Extract parameters from an [DcLbmFit-class](#) object

**Usage**

```
## S4 method for signature 'DcLbmFit'
coef(object)
```

**Arguments**

object                a [DcLbmFit-class](#)

**Value**

a list with the model parameters estimates (MAP), the fields are:

- 'pirows': row cluster proportions
- 'picols': col cluster proportions
- 'thetakl': between clusters connection probabilities (matrix of size Krow x Kcol),
- 'gammarrow': rows degree correction parameters (size Nrows),
- 'gammacol': cols degree correction parameters (size Ncols),

---

coef, DcSbmFit-method    *Extract parameters from an [DcSbmFit-class](#) object*

---

## Description

Extract parameters from an [DcSbmFit-class](#) object

## Usage

```
## S4 method for signature 'DcSbmFit'  
coef(object)
```

## Arguments

object            a [DcSbmFit-class](#)

## Details

in case of undirected graph

## Value

a list with the model parameters estimates (MAP), the fields are the following for "directed" models :

- 'pi': cluster proportions
- 'thetakl': between cluster normalized connection intensities (matrix of size  $K \times K$ ),
- gammain: node in-degree correction parameter
- gammaout: node out-degree correction parameter

And as follow for un-directed models : #'

- 'pi': cluster proportions
- 'thetakl': between cluster normalized connection intensities (matrix of size  $K \times K$ ),
- gamma: node degree correction parameter

---

coef,DiagGmmFit-method

*Extract mixture parameters from [DiagGmmFit-class](#) object*

---

### Description

Extract mixture parameters from [DiagGmmFit-class](#) object

### Usage

```
## S4 method for signature 'DiagGmmFit'  
coef(object)
```

### Arguments

object            a [DiagGmmFit-class](#)

### Value

a list with the mixture parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'muk': cluster means
- 'Sigmak': cluster co-variance matrices

---

coef,GmmFit-method

*Extract mixture parameters from [GmmFit-class](#) object*

---

### Description

Extract mixture parameters from [GmmFit-class](#) object

### Usage

```
## S4 method for signature 'GmmFit'  
coef(object)
```

### Arguments

object            a [GmmFit-class](#)

### Value

a list with the mixture parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'muk': cluster means
- 'Sigmak': cluster co-variance matrices



---

coef,IclFit-method      *Extract parameters from an [IclFit-class](#) object*

---

### Description

Extract parameters from an [IclFit-class](#) object

### Usage

```
## S4 method for signature 'IclFit'  
coef(object)
```

### Arguments

object                  a [IclFit-class](#)

### Details

The results depends of the used model, in case the method is not yet implemented for a model, this generic method will be used. Which will return the obs\_stats slot of the model.

### Value

a list with the model parameters estimates (MAP)

---

coef,LcaFit-method      *Extract parameters from an [LcaFit-class](#) object*

---

### Description

Extract parameters from an [LcaFit-class](#) object

### Usage

```
## S4 method for signature 'LcaFit'  
coef(object)
```

### Arguments

object                  a [LcaFit-class](#)

### Value

a list with the model parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'thetav': cluster profile probabilities (list of matrix of size  $K \times D_v$ ),

---

coef,MoMFit-method      *Extract parameters from an MoMFit-class object*

---

### Description

Extract parameters from an MoMFit-class object

### Usage

```
## S4 method for signature 'MoMFit'
coef(object)
```

### Arguments

object                  a MoMFit-class

### Value

a list with the model parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'thetak': cluster profile probabilities (matrix of size K x D),

---

coef,MoRFit-method      *Extract mixture parameters from MoRFit-class object using MAP estimation*

---

### Description

Extract mixture parameters from MoRFit-class object using MAP estimation

### Usage

```
## S4 method for signature 'MoRFit'
coef(object)
```

### Arguments

object                  a MoRFit-class

### Value

a list with the mixture parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'A': cluster regression matrix
- 'Sigma\_k': cluster noise co-variance matrices

---

coef, MultSbmFit-method

*Extract parameters from an [MultSbmFit-class](#) object*

---

**Description**

Extract parameters from an [MultSbmFit-class](#) object

**Usage**

```
## S4 method for signature 'MultSbmFit'  
coef(object)
```

**Arguments**

object            a [MultSbmFit-class](#)

**Value**

a list with the model parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'thetak1': cluster profile probabilities (array of size  $K \times K \times D$ ),

---

coef, SbmFit-method

*Extract parameters from an [SbmFit-class](#) object*

---

**Description**

Extract parameters from an [SbmFit-class](#) object

**Usage**

```
## S4 method for signature 'SbmFit'  
coef(object)
```

**Arguments**

object            a [SbmFit-class](#)

**Value**

a list with the model parameters estimates (MAP), the fields are:

- 'pi': cluster proportions
- 'thetak1': between clusters connections probabilities (matrix of size  $K \times K$ )

---

CombinedModels

*Combined Models classes*

---

### Description

An S4 class to represent a combined clustering models, where several models are used to model different datasets. A conditional independence assumption between the view knowing the cluster is made.

### Usage

```
CombinedModels(models, alpha = 1)
```

### Arguments

models	a named list of DlvPrior's object
alpha	Dirichlet prior parameter over the cluster proportions (default to 1)

### Details

The filed name in the models list must match the name of the list use to provide the datasets to cluster together.

### Value

a CombinedModels-class object

### See Also

[CombinedModelsFit-class](#), [CombinedModelsPath-class](#)

Other DlvModels: [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

### Examples

```
CombinedModels(models = list(continuous = GmmPrior(), discrete = LcaPrior()))
```

---

 CombinedModelsFit-class

*Combined Models fit results class*


---

### Description

An S4 class to represent a fit of a degree corrected stochastic block model for co\_clustering, extend [IclFit-class](#).

### Slots

model a [DcSbm-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

cl a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

move\_mat binary matrix which store move constraints

train\_hist data.frame with training history information (details depends on the training procedure)

### See Also

[extractSubModel](#), [CombinedModelsPath](#), [character-method](#)

---

 CombinedModelsPath-class

*Combined Models hierarchical fit results class*


---

### Description

An S4 class to represent a hierarchical fit of a degree corrected stochastic block model, extend [IclPath-class](#).

### Slots

model a [DcSbm-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

cl a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

path a list of size K-1 with each part of the path described by:

- icl1: icl value reach with this solution for alpha=1
- logalpha: log(alpha) value were this solution is better than its parent
- K: number of clusters
- cl: vector of cluster indexes
- k,l: index of the cluster that were merged at this step
- merge\_mat: lower triangular matrix of delta icl values
- obs\_stats: a list with the elements:

logalpha value of log(alpha)

ggtree data.frame with complete merge tree for easy plotting with ggplot2

tree numeric vector with merge tree tree[i] contains the index of i father

train\_hist data.frame with training history information (details depends on the training procedure)

### See Also

[extractSubModel](#), [CombinedModelsPath](#), [character-method](#)

---

cut,DcLbmPath-method *Method to cut a DcLbmPath solution to a desired number of cluster*

---

### Description

This method take a [DcLbmPath-class](#) and an integer K and return the solution from the path with K clusters

### Usage

```
## S4 method for signature 'DcLbmPath'
cut(x, K)
```

### Arguments

x	A an <a href="#">DcLbmPath-class</a> solution
K	Desired number of cluster

### Value

an [IclPath-class](#) object with the desired number of cluster

---

cut,IclPath-method      *Generic method to cut a path solution to a desired number of cluster*

---

### Description

This method take a [IclPath-class](#) object and an integer K and return the solution from the path with K clusters

### Usage

```
## S4 method for signature 'IclPath'
cut(x, K)
```

### Arguments

x	A an IclPath solution
K	Desired number of cluster

### Value

an [IclPath-class](#) object with the desired number of cluster

---

DcLbm      *Degree Corrected Latent Block Model for bipartite graph class*

---

### Description

An S4 class to represent a degree corrected stochastic block model for co\_clustering of bipartite graph. Such model can be used to cluster graph vertex, and model a bipartite graph adjacency matrix  $X$  with the following generative model :

$$\begin{aligned} \pi &\sim \text{Dirichlet}(\alpha) \\ Z_i^r &\sim \mathcal{M}(1, \pi^r) \\ Z_j^c &\sim \mathcal{M}(1, \pi^c) \\ \theta_{kl} &\sim \text{Exponential}(p) \\ \gamma_i^r &\sim \mathcal{U}(S_k) \\ \gamma_i^c &\sim \mathcal{U}(S_l) \\ X_{ij} | Z_{ik}^c, Z_{jl}^r = 1 &\sim \mathcal{P}(\gamma_i^r \theta_{kl} \gamma_j^c) \end{aligned}$$

The individuals parameters  $\gamma_i^r, \gamma_j^c$  allow to take into account the node degree heterogeneity. These parameters have uniform priors over simplex  $S_k$ . These classes mainly store the prior parameters value  $\alpha, p$  of this generative model. The DcLbm-class must be used when fitting a simple Diagonal Gaussian Mixture Model whereas the DcLbmPrior-class must be sued when fitting a [CombinedModels-class](#).

**Usage**

```
DcLbmPrior(p = NaN)
```

```
DcLbm(alpha = 1, p = NaN)
```

**Arguments**

**p** Exponential prior parameter (default to Nan, in this case p will be estimated from data as the average intensities of X)

**alpha** Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a DcLbmPrior-class  
a DcLbm-class object

**See Also**

[DcLbmFit-class](#), [DcLbmPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
DcLbmPrior()
DcLbmPrior(p = 0.7)
DcLbm()
DcLbm(p = 0.7)
```

---

DcLbmFit-class

*Degree corrected Latent Block Model fit results class*

---

**Description**

An S4 class to represent a fit of a degree corrected stochastic block model for co\_clustering, extend [IclFit-class](#).

**Slots**

**model** a [DcLbm-class](#) object to store the model fitted

**name** generative model name

**icl** icl value of the fitted model

**K** number of extracted clusters over row and columns

**Krow** number of extracted row clusters

**Kcol** number of extracted column clusters



`cl` a numeric vector with row and columns cluster indexes  
`obs_stats` a list with the following elements:

- `counts`: numeric vector of size  $K$  with number of elements in each clusters
- `din`: numeric vector of size  $K$  which store the sums of in-degrees for each clusters
- `dout`: numeric vector of size  $K$  which store the sums of out-degrees for each clusters
- `x_counts`: matrix of size  $K \times K$  with the number of links between each pair of clusters
- `co_x_counts`: matrix of size  $K_{row} \times K_{col}$  with the number of links between each pair of row and column cluster

`clrow` a numeric vector with row cluster indexes  
`clcol` a numeric vector with column cluster indexes  
`Nrow` number of rows  
`Ncol` number of columns  
`move_mat` binary matrix which store move constraints  
`train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[coef, DcLbmFit-method](#)

---

DcLbmPath-class

*Degree corrected Latent Block Model hierarchical fit results class*

---

**Description**

An S4 class to represent a fit of a degree corrected stochastic block model for co\_clustering, extend [IclPath-class](#).

**Slots**

`model` a [DcLbm-class](#) object to store the model fitted  
`name` generative model name  
`icl` icl value of the fitted model  
`K` number of extracted clusters over row and columns  
`Krow` number of extracted row clusters  
`Kcol` number of extracted column clusters  
`cl` a numeric vector with row and columns cluster indexes  
`obs_stats` a list with the following elements:

- `counts`: numeric vector of size  $K$  with number of elements in each clusters
- `din`: numeric vector of size  $K$  which store the sums of in-degrees for each clusters
- `dout`: numeric vector of size  $K$  which store the sums of out-degrees for each clusters

- `x_counts`: matrix of size  $K \times K$  with the number of links between each pair of clusters
- `co_x_counts`: matrix of size  $K_{row} \times K_{col}$  with the number of links between each pair of row and column cluster

`clrow` a numeric vector with row cluster indexes

`clcol` a numeric vector with column cluster indexes

`Nrow` number of rows

`Ncol` number of columns

`path` a list of size  $K-1$  with each part of the path described by:

- `icl1`: icl value reach with this solution for  $\alpha=1$
- `logalpha`:  $\log(\alpha)$  value were this solution is better than its parent
- `K`: number of clusters
- `cl`: vector of cluster indexes
- `k,l`: index of the cluster that were merged at this step
- `merge_mat`: lower triangular matrix of delta icl values
- `obs_stats`: a list with the elements:
  - `counts`: numeric vector of size  $K$  with number of elements in each clusters
  - `din`: numeric vector of size  $K$  which store the sums of in-degrees for each clusters
  - `dout`: numeric vector of size  $K$  which store the sums of out-degrees for each clusters
  - `x_counts`: matrix of size  $K \times K$  with the number of links between each pair of clusters
  - `co_x_counts`: matrix of size  $K_{row} \times K_{col}$  with the number of links between each pair of row and column cluster

`logalpha` value of  $\log(\alpha)$

`ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`

`tree` numeric vector with merge tree `tree[i]` contains the index of  $i$  father

`ggtreerow` data.frame with complete merge tree of row clusters for easy plotting with `ggplot2`

`ggtreecol` data.frame with complete merge tree of column clusters for easy plotting with `ggplot2`

`train_hist` data.frame with training history information (details depends on the training procedure)

### See Also

[plot,DcLbmPath,missing-method](#)

**Description**

An S4 class to represent a Degree Corrected Stochastic Block Model. Such model can be used to cluster graph vertex, and model a square adjacency matrix  $X$  with the following generative model :

$$\begin{aligned}\pi &\sim \text{Dirichlet}(\alpha) \\ Z_i &\sim \mathcal{M}(1, \pi) \\ \theta_{kl} &\sim \text{Exponential}(p) \\ \gamma_i^+, \gamma_i^- &\sim \mathcal{U}(S_k) \\ X_{ij} | Z_{ik} Z_{jl} = 1 &\sim \mathcal{P}(\gamma_i^+ \theta_{kl} \gamma_j^-)\end{aligned}$$

The individuals parameters  $\gamma_i^+, \gamma_i^-$  allow to take into account the node degree heterogeneity. These parameters have uniform priors over the simplex  $S_k$  ie.  $\sum_{i:z_{ik}=1} \gamma_i^+ = 1$ . These classes mainly store the prior parameters value  $\alpha, p$  of this generative model. The DcSbm-class must be used when fitting a simple Degree Corrected Stochastic Block Model whereas the DcSbmPrior-class must be used when fitting a [CombinedModels-class](#).

**Usage**

```
DcSbmPrior(p = NaN, type = "guess")
```

```
DcSbm(alpha = 1, p = NaN, type = "guess")
```

**Arguments**

p	Exponential prior parameter (default to NaN, in this case p will be estimated from data as the mean connection probability)
type	define the type of networks (either "directed", "undirected" or "guess", default to "guess")
alpha	Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a DcSbmPrior-class object

a DcSbm-class object

**See Also**

[DcSbmFit-class](#), [DcSbmPath-class](#)

Other DlvModels: [CombinedModels](#), [DcLbm](#), [DiagGmm](#), [DlvPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
DcSbmPrior()
DcSbmPrior(type = "undirected")
DcSbm()
DcSbm(type = "undirected")
```

---

DcSbmFit-class

*Degree Corrected Stochastic Block Model fit results class*


---

### Description

An S4 class to represent a fit of a degree corrected stochastic block model for co\_clustering, extend [IclFit-class](#).

### Slots

model a [DcSbm-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

c1 a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

- counts: numeric vector of size K with number of elements in each clusters
- din: numeric vector of size K which store the sums of in-degrees for each clusters
- dout: numeric vector of size K which store the sums of out-degrees for each clusters
- x\_counts: matrix of size K\*K with the number of links between each pair of clusters

obs\_stats\_cst a list with the following elements:

- din\_node: node in-degree, a vector of size N
- dout\_node: node in-degree vector of size N

move\_mat binary matrix which store move constraints

train\_hist data.frame with training history information (details depends on the training procedure)

### See Also

[coef, DcSbmFit-method](#)

---

DcSbmPath-class

*Degree Corrected Stochastic Block Model hierarchical fit results class*


---

### Description

An S4 class to represent a hierarchical fit of a degree corrected stochastic block model, extend [IclPath-class](#).

**Slots**

`model` a `DcSbm-class` object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`cl` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size `K` with number of elements in each clusters
- `din`: numeric vector of size `K` which store the sums of in-degrees for each clusters
- `dout`: numeric vector of size `K` which store the sums of out-degrees for each clusters
- `x_counts`: matrix of size `K*K` with the number of links between each pair of clusters

`path` a list of size `K-1` with each part of the path described by:

- `icl1`: icl value reach with this solution for `alpha=1`
- `logalpha`: `log(alpha)` value were this solution is better than its parent
- `K`: number of clusters
- `cl`: vector of cluster indexes
- `k,l`: index of the cluster that were merged at this step
- `merge_mat`: lower triangular matrix of delta icl values
- `obs_stats`: a list with the elements:
  - `counts`: numeric vector of size `K` with number of elements in each clusters
  - `din`: numeric vector of size `K` which store the sums of in-degrees for each clusters
  - `dout`: numeric vector of size `K` which store the sums of out-degrees for each clusters
  - `x_counts`: matrix of size `K*K` with the number of links between each pair of clusters

`logalpha` value of `log(alpha)`

`ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`

`tree` numeric vector with merge tree `tree[i]` contains the index of `i` father

`train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[plot,DcSbmFit,missing-method](#)

DiagGmm

*Diagonal Gaussian Mixture Model Prior description class***Description**

An S4 class to represent a multivariate diagonal Gaussian mixture model. The model corresponds to the following generative model:

$$\begin{aligned}\pi &\sim \text{Dirichlet}(\alpha) \\ Z_i &\sim \mathcal{M}(1, \pi) \\ \lambda_k^{(d)} &\sim \mathcal{G}(\kappa, \beta) \\ \mu_k^{(d)} &\sim \mathcal{N}(\mu, (\tau\lambda_k)^{-1}) \\ X_{i.}|Z_{ik} = 1 &\sim \mathcal{N}(\mu_k, \lambda_k^{-1})\end{aligned}$$

with  $\mathcal{G}(\kappa, \beta)$  the Gamma distribution with shape parameter  $\kappa$  and rate parameter  $\beta$ . These classes mainly store the prior parameters value  $(\alpha, \tau, \kappa\beta, \mu)$  of this generative model. The `DiagGmm`-class must be used when fitting a simple Diagonal Gaussian Mixture Model whereas the `DiagGmmPrior`-class must be used when fitting a [CombinedModels-class](#).

**Usage**

```
DiagGmmPrior(tau = 0.01, kappa = 1, beta = NaN, mu = NaN)
```

```
DiagGmm(alpha = 1, tau = 0.01, kappa = 1, beta = NaN, mu = NaN)
```

**Arguments**

tau	Prior parameter (inverse variance), (default 0.01)
kappa	Prior parameter (gamma shape), (default to 1)
beta	Prior parameter (gamma rate), (default to NaN, in this case beta will be estimated from data as 0.1 time the mean of X columns variances)
mu	Prior for the means (vector of size D), (default to NaN, in this case mu will be estimated from data as the mean of X)
alpha	Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a `DiagGmmPrior`-class object

a `DiagGmm`-class object

**References**

Bertoletti, Marco & Friel, Nial & Rastelli, Riccardo. (2014). Choosing the number of clusters in a finite mixture model using an exact Integrated Completed Likelihood criterion. METRON. 73. 10.1007/s40300-015-0064-5. #'

**See Also**

[DiagGmmFit-class](#), [DiagGmmPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DlvMPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
DiagGmmPrior()
DiagGmmPrior(tau = 0.1)
DiagGmm()
DiagGmm(tau = 0.1)
```

---

[DiagGmmFit-class](#)

*Diagonal Gaussian mixture model fit results class*

---

**Description**

An S4 class to represent a fit of a multivariate diagonal Gaussian mixture model, extend [IclFit-class](#).

**Slots**

`model` a [DiagGmm-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`c1` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size `K` with number of elements in each clusters
- `regs`: list of size `$K$` with statistics for each clusters

`move_mat` binary matrix which store move constraints

`train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[coef](#), [DiagGmmFit-method](#)

---

DiagGmmPath-class      *Diagonal Gaussian mixture model hierarchical fit results class*

---

## Description

An S4 class to represent a hierarchical fit of a diagonal gaussian mixture model, extend [IclPath-class](#).

## Slots

`model` a [DiagGmm-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`c1` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size `K` with number of elements in each clusters
- `regs`: list of size `$K$` with statistics for each clusters

`path` a list of size `K-1` with each part of the path described by:

- `icl1`: icl value reach with this solution for `alpha=1`
- `logalpha`: `log(alpha)` value were this solution is better than its parent
- `K`: number of clusters
- `c1`: vector of cluster indexes
- `k,l`: index of the cluster that were merged at this step
- `merge_mat`: lower triangular matrix of delta icl values
- `obs_stats`: a list with the following elements:
  - `counts`: numeric vector of size `K` with number of elements in each clusters
  - `regs`: list of size `$K$` with statistics for each clusters

`logalpha` value of `log(alpha)`

`ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`

`tree` numeric vector with merge tree `tree[i]` contains the index of `i` father

`train_hist` data.frame with training history information (details depends on the training procedure)

## See Also

[plot,DiagGmmFit,missing-method](#)



---

DlvmCoPrior-class      *Abstract class to represent a generative model for co-clustering*

---

**Description**

An S4 class to represent an abstract generative model

**Slots**

alpha a numeric vector of length 1 which define the parameters of the Dirichlet over the cluster proportions (default to 1)

---

DlvmPrior-class      *Abstract class to represent a generative model for clustering*

---

**Description**

An S4 class to represent an abstract generative model

**Slots**

alpha a numeric vector of length 1 which define the parameters of the Dirichlet over the cluster proportions (default to 1)

**See Also**

Other DlvmModels: [CombinedModels](#), [Dclbm](#), [DcSbm](#), [DiagGmm](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

---

extractSubModel      *Extract a part of a [CombinedModelsPath-class](#) object*

---

**Description**

Extract a part of a [CombinedModelsPath-class](#) object

**Usage**

```
extractSubModel(sol, sub_model_name)
```

```
## S4 method for signature 'CombinedModelsPath,character'
extractSubModel(sol, sub_model_name)
```

**Arguments**

- `sol` an `CombinedModelsPath`-class object
- `sub_model_name` a string which specify the part of the model to extract. Note that the name must correspond to the one of the names used in the list of models during the origin call to `greed`.

**Value**

a `IclFit`-class object of the relevant class

**Methods (by class)**

- `sol = CombinedModelsPath, sub_model_name = character`: `CombinedModelsPath` method

---

fashion	<i>Fashion mnist dataset</i>
---------	------------------------------

---

**Description**

Zalando fashionmnist dataset, sample of 1 000 Zalando's article images from the test set.

**Usage**

```
data(fashion)
```

**Format**

An object of class `matrix` with a random sample of 1000 images (one per rows) extracted from the fashionmnist dataset.

**Source**

<https://github.com/zalandoresearch/fashion-mnist>

**References**

Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf (2017) ([arXiv:1708.07747](https://arxiv.org/abs/1708.07747)).

**Examples**

```
data(fashion)
```

---

Fifa	<i>Fifa data</i>
------	------------------

---

**Description**

A random sample of 6000 players from the FIFA videogame with various statistics on all player ranging from position, cost in the game, capacity in offense/defense, speed, etc. Two columns `pos_x`, `pos_y` with average player possible positions (in opta coordiantes) were derived from the raw data. was also u.

**Usage**

```
data(Fifa)
```

**Format**

An R `data.frame` with columns containing each of the descriptive statistics of a player.

**Source**

[https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset?select=players\\_20.csv](https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset?select=players_20.csv)

**Examples**

```
data(Fifa)
```

---

Football	<i>American College football network dataset</i>
----------	--

---

**Description**

Network of American football games between Division IA colleges during regular season Fall 2000.

**Usage**

```
data(Football)
```

**Format**

An object of class `list` with two fields;

**X** network adjacency matrix as a `sparseMatrix` of size 115x115

**label** vector of teams conferences of size 115 with the following encoding (0 = Atlantic Coast, 1 = Big East, 2 = Big Ten, 3 = Big Twelve, 4 = Conference USA, 5 = Independents, 6 = Mid-American, 7 = Mountain West, 8 = Pacific Ten, 9 = Southeastern, 10 = Sun Belt, 11 = Western Athletic)

**Source**

[M. E. J. Newman Network datasets](#)

**References**

M. Girvan and M. E. J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002) ([PNAS](#)).

**Examples**

```
data(Football)
```

---

Genetic-class

*Genetic optimization algorithm*

---

**Description**

An S4 class to represent a genetic algorithm (extends [Alg-class](#) class).

**Usage**

```
Genetic(pop_size = 100, nb_max_gen = 20, prob_mutation = 0.25, sel_frac = 0.75)
```

**Arguments**

pop_size	size of the solutions populations (default to 10)
nb_max_gen	maximal number of generation to produce (default to 4)
prob_mutation	probability of mutation (default to 0.25)
sel_frac	fraction of best solutions selected for crossing (default to 0.75)

**Value**

a Genetic-class object

**Functions**

- Genetic: Genetic algorithm class constructor

**Slots**

pop_size	size of the solutions populations (default to 10)
nb_max_gen	maximal number of generation to produce (default to 4)
prob_mutation	probability of mutation (default to 0.25)
sel_frac	fraction of best solutions selected for crossing (default to 0.75)

**Examples**

```
Genetic()
Genetic(pop_size = 500)
```

Gmm

*Gaussian Mixture Model Prior description class***Description**

An S4 class to represent a multivariate Gaussian mixture model. The model corresponds to the following generative model:

$$\begin{aligned}\pi &\sim \text{Dirichlet}(\alpha) \\ Z_i &\sim \mathcal{M}(1, \pi) \\ V_k &\sim \mathcal{W}(\varepsilon^{-1}, n_0) \\ \mu_k &\sim \mathcal{N}(\mu, (\tau V_k)^{-1}) \\ X_i | Z_{ik} = 1 &\sim \mathcal{N}(\mu_k, V_k^{-1})\end{aligned}$$

with  $\mathcal{W}(\varepsilon^{-1}, n_0)$  the Wishart distribution. The `Gmm`-class must be used when fitting a simple Gaussian Mixture Model whereas the `GmmPrior`-class must be used when fitting a [CombinedModels](#)-class.

**Usage**

```
GmmPrior(tau = 0.01, N0 = NaN, mu = NaN, epsilon = NaN)
```

```
Gmm(tau = 0.01, N0 = NaN, mu = NaN, epsilon = NaN, alpha = 1)
```

**Arguments**

tau	Prior parameter (inverse variance) default 0.01
N0	Prior parameter (pseudo count) should be > number of features (default to NaN, in this case it will be estimated from data as the number of columns of X)
mu	Prior parameters for the means (vector of size D), (default to NaN, in this case mu will be estimated from the data and will be equal to the mean of X)
epsilon	Prior parameter co-variance matrix prior (matrix of size D x D), (default to a matrix of NaN, in this case epsilon will be estimated from data and will corresponds to 0.1 times a diagonal matrix with the variances of the X columns)
alpha	Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a `GmmPrior`-class object

a `Gmm`-class object

**References**

Bertoletti, Marco & Friel, Nial & Rastelli, Riccardo. (2014). Choosing the number of clusters in a finite mixture model using an exact Integrated Completed Likelihood criterion. METRON. 73. 10.1007/s40300-015-0064-5.

**See Also**

[GmmFit-class](#), [GmmPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
GmmPrior()
GmmPrior(tau = 0.1)
Gmm()
Gmm(tau = 0.1, alpha = 0.5)
```

---

GmmFit-class

*Gaussian mixture model fit results class*

---

**Description**

An S4 class to represent a fit of a multivariate mixture of regression model, extend [IclFit-class](#).

**Slots**

`model` a [GmmPrior-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`cl` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size K with number of elements in each clusters
- `regs`: list of size  $K \times K$  with statistics for each clusters

`move_mat` binary matrix which store move constraints

`train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[coef](#), [GmmFit-method](#)

---

gmmpairs	<i>Make a matrix of plots with a given data and gmm fitted parameters</i>
----------	---

---

**Description**

Make a matrix of plots with a given data and gmm fitted parameters with ellipses.

**Usage**

```
gmmpairs(sol, X)
```

**Arguments**

sol	a <a href="#">GmmFit-class</a> or <a href="#">DiagGmmFit-class</a>
X	the data used for the fit a data.frame or matrix.

**Value**

a [ggplot2](#) graphic

---

GmmPath-class	<i>Gaussian mixture model hierarchical fit results class</i>
---------------	--

---

**Description**

An S4 class to represent a hierarchical fit of a gaussian mixture model, extend [IclPath-class](#).

**Slots**

model	a <a href="#">GmmPrior-class</a> object to store the model fitted
name	generative model name
icl	icl value of the fitted model
K	number of extracted clusters over row and columns
c1	a numeric vector with row and columns cluster indexes
obs_stats	a list with the following elements: <ul style="list-style-type: none"> <li>• counts: numeric vector of size K with number of elements in each clusters</li> <li>• gmm: list of size \$K\$ with statistics for each clusters</li> </ul>
path	a list of size K-1 with each part of the path described by: <ul style="list-style-type: none"> <li>• icl1: icl value reach with this solution for alpha=1</li> <li>• logalpha: log(alpha) value were this solution is better than its parent</li> <li>• K: number of clusters</li> <li>• k,l: index of the cluster that were merged at this step</li> </ul>

- `merge_mat`: lower triangular matrix of delta icl values
- `obs_stats`: a list with the following elements:
  - `counts`: numeric vector of size `K` with number of elements in each clusters
  - `gmm`: list of size `$K$` with statistics for each clusters

`logalpha` value of  $\log(\alpha)$

`ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`

`tree` numeric vector with merge tree `tree[i]` contains the index of `i` father

`train_hist` data.frame with training history information (details depends on the training procedure)

### See Also

[plot](#), [GmmFit](#), [missing-method](#)

---

greed

*Model based hierarchical clustering*

---

### Description

This function is the main function for fitting DLVMs with `greed`. In the simplest case you may only provide a dataset and `greed` will find a suitable one. The accepted classes for `X` depends on the generative used which can be specified with the `model` argument. See the [DlvmPrior-class](#) and the derived classes for details.

`greed` enables the clustering of networks and count data matrix with different models. Model selection and clustering are performed in combination by optimizing the Integrated Classification Likelihood. Optimization is performed thanks to a combination of greedy local search and a genetic algorithm. The main entry point is the `greed` function to perform the clustering, which is documented below. The package also provides sampling functions for all the implemented DLVMs.

### Usage

```
greed(X, model = find_model(X), K = 20, alg = Hybrid(), verbose = FALSE)
```

### Arguments

<code>X</code>	data to cluster either a data.frame, a matrix, an array, ... depending on the used generative model
<code>model</code>	a generative model to fit such as <a href="#">Gmm</a> , <a href="#">Sbm</a> , ..
<code>K</code>	initial number of cluster
<code>alg</code>	an optimization algorithm of class <a href="#">Alg-class</a> such as <a href="#">Hybrid-class</a> (default), <a href="#">Multistarts-class</a> , <a href="#">Seed-class</a> or <a href="#">Genetic-class</a>
<code>verbose</code>	boolean value for verbose mode



**Value**

an `IclPath-class` object

**See Also**

Other `DlvmModels`: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvmPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#)

**Examples**

```
sbm <- rsbm(50, c(0.5, 0.5), diag(2) * 0.1 + 0.01)
sol <- greed(sbm$x, model = Sbm())
table(sbm$c1, clustering(sol))
```

---

H

*Compute the entropy of a discrete sample*

---

**Description**

Compute the entropy of a discrete sample

**Usage**

```
H(c1)
```

**Arguments**

`c1`                    vector of discrete labels

**Value**

the entropy of the sample

**Examples**

```
c1 <- sample(2, 500, replace = TRUE)
H(c1)
```

---

Hybrid-class	<i>Hybrid optimization algorithm</i>
--------------	--------------------------------------

---

**Description**

An S4 class to represent an hybrid genetic/greedy algorithm (extends [Alg-class](#) class).

**Usage**

```
Hybrid(pop_size = 20, nb_max_gen = 10, prob_mutation = 0.25, Kmax = 100)
```

**Arguments**

pop_size	size of the solutions populations (default to 20)
nb_max_gen	maximal number of generation to produce (default to 10)
prob_mutation	mutation probability (default to 0.25)
Kmax	maximum number of clusters (default to 100)

**Value**

a Hybrid-class object

**Functions**

- Hybrid: Hybrid algorithm class constructor

**Slots**

pop_size	size of the solutions populations (default to 20)
nb_max_gen	maximal number of generation to produce (default to 10)
prob_mutation	mutation probability (default to 0.25)
Kmax	maximum number of clusters (default to 100)

**Examples**

```
Hybrid()  
Hybrid(pop_size = 100)
```

---

ICL	<i>Generic method to extract the ICL value from an <code>IclFit</code>-class object</i>
-----	---

---

**Description**

This method take a `IclFit`-class object and return its ICL score.

**Usage**

```
ICL(fit)

## S4 method for signature 'IclFit'
ICL(fit)
```

**Arguments**

`fit` an `IclFit` solution

**Value**

The ICL value achieved

**Methods (by class)**

- `IclFit`: `IclFit` method

---

<code>IclFit</code> -class	<i>Abstract class to represent a clustering result</i>
----------------------------	--

---

**Description**

An S4 abstract class to represent an icl fit of a clustering model.

**Slots**

`K` a numeric vector of length 1 which correspond to the number of clusters  
`icl` a numeric vector of length 1 which store the the icl value  
`c1` a numeric vector of length N which store the clusters labels  
`obs_stats` a list to store the observed statistics of the model needed to compute ICL.  
`obs_stats_cst` a list to store the observed statistics of the model that do not depend on the clustering.  
`move_mat` binary matrix which store move constraints  
`train_hist` a data.frame to store training history (format depends on the used algorithm used).  
`name` generative model name

---

IclPath-class

*Abstract class to represent a hierarchical clustering result*


---

**Description**

An S4 class to represent a hierarchical path of solution.

**Slots**

path a list of merge moves describing the hierarchy of merge followed to complete totally the merge path.

tree a tree representation of the merges.

ggtree a data.frame for easy plotting of the dendrogram

logalpha a numeric value which corresponds to the starting value of  $\log(\alpha)$ .

---

Jazz

*Jazz musicians network dataset*


---

**Description**

List of edges of the network of Jazz musicians.

**Usage**

```
data(Jazz)
```

**Format**

An object of class `sparseMatrix` with the network adjacency matrix.

**Source**

[A. Arena Network datasets](#)

**References**

P.Gleiser and L. Danon , Community Structure in jazz, Adv. Complex Syst.6, 565 (2003) ([Arxiv](#))

**Examples**

```
data(Jazz)
```

---

K	<i>Generic method to get the number of clusters from an <a href="#">IclFit-class</a> object</i>
---	---

---

**Description**

This method take a [IclFit-class](#) object and return its ICL score.

**Usage**

```
K(fit)

## S4 method for signature 'IclFit'
K(fit)
```

**Arguments**

`fit`                    an IclFit solution

**Value**

The number of clusters

**Methods (by class)**

- `IclFit`: IclFit method

---

Lca	<i>Latent Class Analysis Model Prior class</i>
-----	--

---

**Description**

An S4 class to represent a Latent Class Analysis model Such model can be used to cluster a data.frame  $X$  with several columns of factors with the following generative model :

$$\begin{aligned} \pi &\sim \text{Dirichlet}(\alpha), \\ \forall k, \forall j, \quad \theta_{kj} &\sim \text{Dirichlet}_{d_j}(\beta), \\ Z_i &\sim \mathcal{M}_K(1, \pi), \\ \forall j = 1, \dots, p, \quad X_{ij} | Z_{ik} = 1 &\sim \mathcal{M}_{d_j}(1, \theta_{kj}), \end{aligned}$$

These classes mainly store the prior parameters value  $(\alpha, \beta)$  of this generative model. The `Lca`-class must be used when fitting a simple Latent Class Analysis whereas the `LcaPrior`-class must be used when fitting a [CombinedModels-class](#).

**Usage**

```
LcaPrior(beta = 1)
```

```
Lca(alpha = 1, beta = 1)
```

**Arguments**

beta            Dirichlet prior parameter for all the categorical feature (default to 1)

alpha           Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a LcaPrior-class object

a Lca-class object

**See Also**

[LcaFit-class](#), [LcaPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Gmm](#), [MoM](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
LcaPrior()
LcaPrior(beta = 0.5)
Lca()
Lca(beta = 0.5)
```

---

LcaFit-class

*Latent Class Analysis fit results class*

---

**Description**

An S4 class to represent a fit of a Latent Class Analysis model for categorical data clustering, extend [IclFit-class](#). The original data must be an  $n \times p$  matrix where  $p$  is the number of variables and each variable is encoded as a factor (integer-valued).

**Slots**

model a [Lca-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

c1 a numeric vector with cluster indexes

obs\_stats a list with the following elements:

- counts: numeric vector of size K with number of elements in each clusters
- x\_counts: matrix of size K\*D with the number of occurrences of each modality for each clusters

move\_mat binary matrix which store move constraints

train\_hist data.frame with training history information (details depends on the training procedure)

### See Also

[coef](#), [LcaFit-method](#)

---

LcaPath-class

*Latent Class Analysis hierarchical fit results class*

---

### Description

An S4 class to represent a fit of a Latent Class Analysis model, extend [IclPath-class](#).

### Slots

model a [Lca-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

cl a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

- counts: numeric vector of size K with number of elements in each clusters
- x\_counts: matrix of size K\*D with the number of occurrence of modality word in each clusters

path a list of size K-1 with each part of the path described by:

- icl1: icl value reach with this solution for alpha=1
- logalpha: log(alpha) value were this solution is better than its parent
- K: number of clusters
- cl: vector of cluster indexes
- k,l: index of the cluster that were merged at this step
- merge\_mat: lower triangular matrix of delta icl values
- obs\_stats a list with the following elements:
  - counts: numeric vector of size K with number of elements in each clusters
  - x\_counts: matrix of size K\*D with the number of occurrence of modality word in each clusters

logalpha value of log(alpha)

ggtree data.frame with complete merge tree for easy plotting with ggplot2

tree numeric vector with merge tree tree[i] contains the index of i father

train\_hist data.frame with training history information (details depends on the training procedure)

**See Also**

[plot,LcaFit,missing-method](#)

---

 MI

*Compute the mutual information of two discrete samples*

---

**Description**

Compute the mutual information of two discrete samples

**Usage**

```
MI(c11, c12)
```

**Arguments**

c11            vector of discrete labels  
 c12            vector of discrete labels

**Value**

the mutual information between the two discrete samples

**Examples**

```
c11 <- sample(2, 500, replace = TRUE)
c12 <- sample(2, 500, replace = TRUE)
MI(c11, c12)
```

---

 MoM

*Mixture of Multinomial Model Prior description class*

---

**Description**

An S4 class to represent a Mixture of Multinomial model. Such model can be used to cluster a data matrix  $X$  with the following generative model :

$$\pi \sim \text{Dirichlet}(\alpha)$$

$$Z_i \sim \mathcal{M}(1, \pi)$$

$$\theta_k \sim \text{Dirichlet}(\beta)$$

$$X_{i.} | Z_{ik} = 1 \sim \mathcal{M}(L_i, \theta_k)$$

With  $L_i = \sum_d = 1^D X_{id}$ . These classes mainly store the prior parameters value  $(\alpha, \beta)$  of this generative model. The MoM-class must be used when fitting a simple Mixture of Multinomials whereas the MoMPrior-class must be used when fitting a [CombinedModels-class](#).



**Usage**

```
MoMPrior(beta = 1)
```

```
MoM(alpha = 1, beta = 1)
```

**Arguments**

beta            Dirichlet over vocabulary prior parameter (default to 1)

alpha           Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a MoMPrior-class object

a MoM-class object

**See Also**

[MoMFit-class](#), [MoMPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Gmm](#), [Lca](#), [MoR](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
MoMPrior()
MoMPrior(beta = 0.5)
MoM()
MoM(beta = 0.5)
```

---

MoMFit-class

*Mixture of Multinomial fit results class*

---

**Description**

An S4 class to represent a fit of a degree corrected stochastic block model for co\_clustering, extend [IclFit-class](#).

**Slots**

model a [MoM-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

c1 a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

- counts: numeric vector of size K with number of elements in each clusters

- `x_counts`: matrix of size  $K \times D$  with the number of occurrences of each modality for each clusters
- `move_mat` binary matrix which store move constraints
- `train_hist` data.frame with training history information (details depends on the training procedure)

### See Also

[coef, LcaFit-method](#)

---

MoMPath-class

*Mixture of Multinomial hierarchical fit results class*

---

### Description

An S4 class to represent a fit of a stochastic block model, extend [IclPath-class](#).

### Slots

- `model` a [MoM-class](#) object to store the model fitted
- `name` generative model name
- `icl` icl value of the fitted model
- `K` number of extracted clusters over row and columns
- `cl` a numeric vector with row and columns cluster indexes
- `obs_stats` a list with the following elements:
- `counts`: numeric vector of size  $K$  with number of elements in each clusters
  - `x_counts`: matrix of size  $K \times D$  with the number of occurrence of modality word in each clusters
- `path` a list of size  $K-1$  with each part of the path described by:
- `icl1`: icl value reach with this solution for  $\alpha=1$
  - `logalpha`:  $\log(\alpha)$  value were this solution is better than its parent
  - `K`: number of clusters
  - `cl`: vector of cluster indexes
  - `k,l`: index of the cluster that were merged at this step
  - `merge_mat`: lower triangular matrix of delta icl values
  - `obs_stats` a list with the following elements:
    - `counts`: numeric vector of size  $K$  with number of elements in each clusters
    - `x_counts`: matrix of size  $K \times D$  with the number of occurrence of modality word in each clusters
- `logalpha` value of  $\log(\alpha)$
- `ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`
- `tree` numeric vector with merge tree `tree[i]` contains the index of `i` father
- `train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[plot,LcaFit,missing-method](#)

---

 MoR

---

*Multivariate mixture of regression Prior model description class*


---

**Description**

An S4 class to represent a multivariate mixture of regression model. The model follows [minka-linear](https://tminka.github.io/papers/minka-linear.pdf) . The model corresponds to the following generative model:

$$\pi \sim \text{Dirichlet}(\alpha)$$

$$Z_i \sim \mathcal{M}(1, \pi)$$

$$V_k \sim \mathcal{W}(\varepsilon^{-1}, n_0)$$

$$A_k \sim \mathcal{MN}(0, (V_k)^{-1}, \tau X X^\top)$$

$$Y_{i.} | X_{i.}, A_k, Z_{ik} = 1 \sim \mathcal{N}(A_k x_{i.}, V_k^{-1})$$

with  $\mathcal{W}(\varepsilon^{-1}, n_0)$  the Wishart distribution and  $\mathcal{MN}$  the matrix-normal distribution. The MoR-class must be used when fitting a simple Mixture of Regression whereas the MoRPrior-class must be used when fitting a [CombinedModels-class](#).

**Usage**

```
MoRPrior(formula, tau = 0.001, N0 = NaN, epsilon = as.matrix(NaN))
```

```
MoR(formula, alpha = 1, tau = 0.1, N0 = NaN, epsilon = as.matrix(NaN))
```

**Arguments**

formula	a <a href="#">formula</a> that describe the linear model to use
tau	Prior parameter (inverse variance) default 0.001
N0	Prior parameter (default to NaN, in this case N0 will be fixed equal to the number of columns of Y.)
epsilon	Covariance matrix prior parameter (default to NaN, in this case epsilon will be fixed to a diagonal variance matrix equal to 0.1 time the variance of the regression residuals with only one cluster.)
alpha	Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a MoRPrior-class object

a MoR-class object

**See Also**

[MoRFit-class](#), [MoRPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MultSbm](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
MoRPrior(y ~ x1 + x2)
MoRPrior(y ~ x1 + x2, N0 = 100)
MoRPrior(cbind(y1, y2) ~ x1 + x2, N0 = 100)
MoR(y ~ x1 + x2)
MoR(y ~ x1 + x2, N0 = 100)
MoR(cbind(y1, y2) ~ x1 + x2, N0 = 100)
```

---

MoRFit-class

*Clustering with a multivariate mixture of regression model fit results class*

---

**Description**

An S4 class to represent a fit of a multivariate mixture of regression model, extend [IclFit-class](#).

**Slots**

`model` a [MoR-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`c1` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size K with number of elements in each clusters
- `mvmregs`: list of size  $K \times K$  with statistics for each clusters

`move_mat` binary matrix which store move constraints

`train_hist` dataframe with training history information (details depends on the training procedure)

**See Also**

[coef](#), [MoRFit-method](#)

---

MoRPath-class	<i>Multivariate mixture of regression model hierarchical fit results class</i>
---------------	--

---

### Description

An S4 class to represent a hierarchical fit of a multivariate mixture of regression model, extend [IclPath-class](#).

### Slots

`model` a [MoR-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`c1` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size `K` with number of elements in each clusters
- `mvmregs`: list of size `$K$` with statistics for each clusters

`path` a list of size `K-1` with each part of the path described by:

- `icl1`: icl value reach with this solution for `alpha=1`
- `logalpha`: `log(alpha)` value were this solution is better than its parent
- `K`: number of clusters
- `k,l`: index of the cluster that were merged at this step
- `merge_mat`: lower triangular matrix of delta icl values
- `obs_stats`: a list with the following elements:
  - `counts`: numeric vector of size `K` with number of elements in each clusters
  - `mvregs`: list of size `$K$` with statistics for each clusters

`logalpha` value of `log(alpha)`

`ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`

`tree` numeric vector with merge tree `tree[i]` contains the index of `i` father

`train_hist` data.frame with training history information (details depends on the training procedure)

---

Multistarts-class      *Greedy algorithm with multiple start class*

---

### Description

An S4 class to represent a greedy algorithm with multiple start (extends [Alg-class](#) class).

### Usage

```
Multistarts(nb_start = 10)
```

### Arguments

nb\_start      number of random starts (default to 10)

### Value

a Multistarts-class object

### Functions

- Multistarts: Multistarts algorithm class constructor

### Slots

nb\_start    number of random starts (default to 10)

### Examples

```
Multistarts()
Multistarts(15)
```

---

MultSbm      *Multinomial Stochastic Block Model Prior class*

---

### Description

An S4 class to represent a Multinomial Stochastic Block Model. Such model can be used to cluster multi-layer graph vertex, and model a square adjacency cube  $X$  of size  $N \times N \times M$  with the following generative model :

$$\begin{aligned}\pi &\sim \text{Dirichlet}(\alpha) \\ Z_i &\sim \mathcal{M}(1, \pi) \\ \theta_{kl} &\sim \text{Dirichlet}(\beta) \\ X_{ij.} | Z_{ik} Z_{jl} = 1 &\sim \mathcal{M}(L_{ij}, \theta_{kl})\end{aligned}$$

With  $L_{ij} = \sum_{m=1}^M X_{ijm}$ . These classes mainly store the prior parameters value  $\alpha, \beta$  of this generative model. The MultSbm-class must be used when fitting a simple MultSbm whereas the MultSbmPrior-class must be used when fitting a [CombinedModels-class](#).

**Usage**

```
MultSbmPrior(beta = 1, type = "guess")
```

```
MultSbm(alpha = 1, beta = 1, type = "guess")
```

**Arguments**

beta	Dirichlet prior parameter over Multinomial links
type	define the type of networks (either "directed", "undirected" or "guess", default to "guess"), for undirected graphs the adjacency matrix is supposed to be symmetric.
alpha	Dirichlet prior parameter over the cluster proportions (default to 1)

**Value**

a MultSbmPrior-class object

a MultSbm-class object

**See Also**

[MultSbmFit-class](#), [MultSbmPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [Sbm](#), [greed\(\)](#)

**Examples**

```
MultSbmPrior()
MultSbmPrior(type = "undirected")
MultSbm()
MultSbm(type = "undirected")
```

---

MultSbmFit-class	<i>Multinomial Stochastic Block Model fit results class</i>
------------------	---

---

**Description**

An S4 class to represent a fit of a Multinomial Stochastic Block Model, extend [IclFit-class](#).

**Slots**

model	a <a href="#">MultSbm-class</a> object to store the model fitted
name	generative model name
icl	icl value of the fitted model
K	number of extracted clusters over row and columns
cl	a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

- counts: numeric vector of size K with number of elements in each clusters
- x\_counts: cube of size KxKxM with the number of links between each pair of clusters

move\_mat binary matrix which store move constraints

train\_hist data.frame with training history information (details depends on the training procedure)

### See Also

[coef, MultSbmFit-method](#)

---

MultSbmPath-class      *Multinomial Stochastic Block Model hierarchical fit results class*

---

### Description

An S4 class to represent a hierarchical fit of a Multinomial Stochastic Block Model, extend [IclPath-class](#).

### Slots

model a [MultSbm-class](#) object to store the model fitted

name generative model name

icl icl value of the fitted model

K number of extracted clusters over row and columns

c1 a numeric vector with row and columns cluster indexes

obs\_stats a list with the following elements:

- counts: numeric vector of size K with number of elements in each clusters
- x\_counts: matrix of size KxKxM with the number of links between each pair of clusters

path a list of size K-1 with each part of the path described by:

- icl1: icl value reach with this solution for alpha=1
- logalpha: log(alpha) value were this solution is better than its parent
- K: number of clusters
- cl: vector of cluster indexes
- k,l: index of the cluster that were merged at this step
- merge\_mat: lower triangular matrix of delta icl values
- obs\_stats: a list with the elements:
  - counts: numeric vector of size K with number of elements in each clusters
  - x\_counts: matrix of size KxKxM with the number of links between each pair of clusters

logalpha value of log(alpha)

ggtree data.frame with complete merge tree for easy plotting with ggplot2

tree numeric vector with merge tree tree[i] contains the index of i father

train\_hist data.frame with training history information (details depends on the training procedure)



**See Also**

[plot](#), [MultSbmFit](#), [missing-method](#)

---

mushroom

*Mushroom data*

---

**Description**

Categorical data from UCI Machine Learning Repository describing 8124 mushrooms with 22 phenotype variables. Each mushroom is classified as "edible" or "poisonous" and the goal is to recover the mushroom class from its phenotype.

**Usage**

```
data(mushroom)
```

**Format**

An R data.frame with a variable edibility used as label and 22 categorical variables with no names. More detail on the UCI webpage describing the data.

**Source**

<https://archive.ics.uci.edu/ml/datasets/Mushroom>

**Examples**

```
data(mushroom)
```

---

Ndrangheta

*Ndrangheta mafia covert network dataset*

---

**Description**

Network of co-attendance occurrence attendance of suspected members of the Ndrangheta criminal organization at summits (meetings whose purpose is to make important decisions and/or affiliations, but also to solve internal problems and to establish roles and powers) taking place between 2007 and 2009.

**Usage**

```
data(Ndrangheta)
```

**Format**

An object of class `list` with two fields;

**X** network adjacency matrix as a `matrix` of size 146x146

**node\_meta** data frame of nodes meta information with features :

**Id** id of the node, rownames of network adjacency matrix

**Locale** factor with the locali affiliation of the node , "OUT": Suspects not belonging to La Lombardia, "MISS": Information not available, other Locali Id.

**Role** factor with the type of hierarchical position of the node "MISS": Information not available, "boss": high hierarchical position, "aff": affiliate

**Source**

[ucinetsoftware/datasets/covert-networks](https://ucinetsoftware.com/datasets/covert-networks)

**References**

Extended Stochastic Block Models with Application to Criminal Networks, Sirio Legramanti and Tommaso Rigon and Daniele Durante and David B. Dunson, 2021, ([arXiv:2007.08569](https://arxiv.org/abs/2007.08569)).

**Examples**

```
data(Ndrangheta)
```

---

NewGuinea

*NewGuinea data*

---

**Description**

[NewGuinea](#) a social network of 16 tribes, where two types of interactions were recorded, amounting to either friendship or enmity [read-cultures-1954].

**Usage**

```
data(NewGuinea)
```

**Format**

A binary array of size (16,16,3) the first layer encodes enmity, the second, the friendship relations. The third, no relations between the two tribes.

**Source**

[https://networks.skewed.de/net/new\\_guinea\\_tribes](https://networks.skewed.de/net/new_guinea_tribes)

**References**

Kenneth E. Read, "Cultures of the Central Highlands, New Guinea", Southwestern J. of Anthropology, 10(1):1-43 (1954). DOI: 10.1086/soutjanth.10.1.3629074

**Examples**

```
data(NewGuinea)
```

---

NMI

*Compute the normalized mutual information of two discrete samples*

---

**Description**

Compute the normalized mutual information of two discrete samples

**Usage**

```
NMI(c11, c12)
```

**Arguments**

c11            vector of discrete labels  
c12            vector of discrete labels

**Value**

the normalized mutual information between the two discrete samples

**Examples**

```
c11 <- sample(2, 500, replace = TRUE)  
c12 <- sample(2, 500, replace = TRUE)  
NMI(c11, c12)
```

---

plot,DclbmFit,missing-method

*Plot a [DclbmFit-class](#)*

---

**Description**

Plot a [DclbmFit-class](#)

**Usage**

```
## S4 method for signature 'DclbmFit,missing'  
plot(x, type = "blocks")
```

**Arguments**

- |      |  |
|------|--|
| x    | a <a href="#">DcLbmFit-class</a>   |
| type | a string which specify plot type: <ul style="list-style-type: none"> <li>• 'blocks': plot a block matrix with summarizing connections between row and column clusters</li> <li>• 'nodelink': plot a nodelink diagram of the bipartite graph summarizing connections between row and column clusters</li> </ul> |

**Value**

a [ggplot2](#) graphic

---

plot, DcLbmPath, missing-method  
*Plot a [DcLbmPath-class](#)*

---

**Description**

Plot a [DcLbmPath-class](#)

**Usage**

```
## S4 method for signature 'DcLbmPath,missing'
plot(x, type = "tree")
```

**Arguments**

- |      |   |
|------|---|
| x    | a <a href="#">DcLbmPath-class</a>   |
| type | a string which specify plot type: <ul style="list-style-type: none"> <li>• 'tree': plot a co-dendrogram of rows and columns clusters</li> <li>• 'blocks': plot a block matrix with summarizing connections between row and column clusters</li> <li>• 'biplot': plot a block matrix with summarizing connections between row and column clusters aligned with row and clusters dendrograms</li> <li>• 'nodelink': plot a nodelink diagram of the bipartite graph summarizing connections between row and column clusters</li> </ul> |

**Value**

a [ggplot2](#) graphic

---

plot,DcSbmFit,missing-method

*Plot a [DcSbmFit-class](#) object*

---

### Description

Plot a [DcSbmFit-class](#) object

### Usage

```
## S4 method for signature 'DcSbmFit,missing'  
plot(x, type = "blocks")
```

### Arguments

x	a <a href="#">DcSbmFit-class</a>
type	a string which specify plot type: <ul style="list-style-type: none"><li>• 'blocks': plot a block matrix with summarizing connections between clusters</li><li>• 'nodelink': plot a nodelink diagram of the graph summarizing connections between clusters</li></ul>

### Value

a [ggplot2](#) graphic

### See Also

[plot,IcIPath,missing-method](#)

---

plot,DiagGmmFit,missing-method

*Plot a [DiagGmmFit-class](#) object*

---

### Description

Plot a [DiagGmmFit-class](#) object

### Usage

```
## S4 method for signature 'DiagGmmFit,missing'  
plot(x, type = "marginals")
```

**Arguments**

- x a [DiagGmmFit-class](#)
- type a string which specify plot type:
- 'marginals': plot the marginal densities
  - 'violins': make a violin plot for each clusters and features

**Value**

a [ggplot2](#) graphic

---

plot,GmmFit,missing-method

*Plot a [GmmFit-class](#) object*

---

**Description**

Plot a [GmmFit-class](#) object

**Usage**

```
## S4 method for signature 'GmmFit,missing'  
plot(x, type = "marginals")
```

**Arguments**

- x a [GmmFit-class](#)
- type a string which specify plot type:
- 'marginals': plot the marginal densities
  - 'violins': make a violin plot for each clusters and features

**Value**

a [ggplot2](#) graphic

---

plot,IclPath,missing-method  
*Plot an [IclPath-class](#) object*

---

### Description

Plot an [IclPath-class](#) object

### Usage

```
## S4 method for signature 'IclPath,missing'  
plot(x, type = "tree")
```

### Arguments

x	a <a href="#">IclPath-class</a>
type	a string which specify plot type: <ul style="list-style-type: none"><li>• 'front': plot the extracted front ICL, log(alpha)</li><li>• 'path': plot the evolution of ICL with respect to K</li><li>• 'tree': plot the associated dendrogram</li></ul>

### Value

a [ggplot2](#) graphic

---

plot,LcaFit,missing-method  
*Plot a [LcaFit-class](#) object*

---

### Description

Plot a [LcaFit-class](#) object

### Usage

```
## S4 method for signature 'LcaFit,missing'  
plot(x, type = "marginals")
```

### Arguments

x	a <a href="#">LcaFit-class</a>
type	a string which specify plot type: <ul style="list-style-type: none"><li>• 'blocks': plot a block matrix with summarizing connections between clusters</li></ul>

**Value**

a [ggplot2](#) graphic

---

`plot, MoMFit, missing-method`

*Plot a [MoMFit-class](#) object*

---

**Description**

Plot a [MoMFit-class](#) object

**Usage**

```
## S4 method for signature 'MoMFit,missing'
plot(x, type = "blocks")
```

**Arguments**

<code>x</code>	a <a href="#">MoMFit-class</a>
<code>type</code>	a string which specify plot type: <ul style="list-style-type: none"> <li>'blocks': plot a block matrix with summarizing connections between clusters</li> </ul>

**Value**

a [ggplot2](#) graphic

---

`plot, MultSbmFit, missing-method`

*Plot a [MultSbmFit-class](#) object*

---

**Description**

Plot a [MultSbmFit-class](#) object

**Usage**

```
## S4 method for signature 'MultSbmFit,missing'
plot(x, type = "blocks")
```



**Arguments**

- |      |  |
|------|--|
| x    | a <a href="#">MultSbmFit-class</a>   |
| type | a string which specify plot type: <ul style="list-style-type: none"> <li>• 'blocks': plot a block matrix with summarizing connections between clusters</li> <li>• 'nodelink': plot a nodelink diagram of the graph summarizing connections between clusters</li> </ul> |

**Value**

a [ggplot2](#) graphic

---

plot,SbmFit,missing-method

*Plot a [SbmFit-class](#) object*

---

**Description**

Plot a [SbmFit-class](#) object

**Usage**

```
## S4 method for signature 'SbmFit,missing'
plot(x, type = "blocks")
```

**Arguments**

- |      |  |
|------|--|
| x    | a <a href="#">SbmFit-class</a>   |
| type | a string which specify plot type: <ul style="list-style-type: none"> <li>• 'blocks': plot a block matrix with summarizing connections between clusters</li> <li>• 'nodelink': plot a nodelink diagram of the graph summarizing connections between clusters</li> </ul> |

**Value**

a [ggplot2](#) graphic

**See Also**

[plot,IclPath,missing-method](#)

---

prior *Generic method to extract the prior used to fit IclFit-class object*

---

### Description

This method take a [IclFit-class](#) object and return the prior used.

### Usage

```
prior(fit)
```

```
## S4 method for signature 'IclFit'
prior(fit)
```

### Arguments

fit                    an IclFit solution

### Value

An S4 object describing the prior parameters

### Methods (by class)

- IclFit: IclFit method

---

rdcsbm *Generates graph adjacency matrix using a degree corrected SBM*

---

### Description

rdcsbm returns an adjacency matrix and the cluster labels generated randomly using a Degree Corrected Stochastic Block Model.

### Usage

```
rdcsbm(N, pi, mu, betain, betaout)
```

### Arguments

N                    A numeric value the size of the graph to generate

pi                    A numeric vector of length K with clusters proportions. Must sum up to 1.

mu                    A numeric matrix of dim K x K with the connectivity pattern to generate, elements in [0,1].

betain                A numeric vector of length N which specify the in-degree correction will be normalized per cluster during the generation.

betaout               A numeric vector of length N which specify the out-degree correction will be normalized per cluster during the generation.

**Details**

It takes the sample size, cluster proportions and emission matrix, and as input and sample a graph accordingly together with the clusters labels.

**Value**

A list with fields:

- x: the count matrix as a dgCMatrix
- K: number of generated clusters
- N: number of vertex
- cl: vector of clusters labels
- pi: clusters proportions
- mu: connectivity matrix
- betain: normalized in-degree parameters
- betaout: normalized out-degree parameters

---

 rlbm

*Generate a data matrix using a Latent Block Model*


---

**Description**

rlbm returns the adjacency matrix and the cluster labels generated randomly with a Latent Block Model.

**Usage**

```
rlbm(Nr, Nc, pir, pic, mu)
```

**Arguments**

Nr	desired Number of rows
Nc	desired Number of column
pir	A numeric vector of length Kr with rows clusters proportions (will be normalized to sum up to 1).
pic	A numeric vector of length Kc with columns clusters proportions (will be normalized to sum up to 1).
mu	A numeric matrix of dim Kr x Kc with the connectivity pattern to generate. elements in [0,1].

**Details**

This function takes the desired graph size, cluster proportions and connectivity matrix as input and sample a graph accordingly together with the clusters labels.

**Value**

A list with fields:

- x: the generated data matrix as a dgCMatrix
- clr: vector of row clusters labels
- clc: vector of column clusters labels
- Kr: number of generated row clusters
- Kc: number of generated column clusters
- Nr: number of rows
- Nc: number of column
- pir: row clusters proportions
- pic: column clusters proportions
- mu: connectivity matrix

**Examples**

```
simu <- rlbm(500, 1000, rep(1 / 5, 5), rep(1 / 10, 10), matrix(runif(50), 5, 10))
```

---

rlca

*Generate data from lca model*

---

**Description**

rlca returns a data.frame with factor sampled from an lca model

**Usage**

```
rlca(N, pi, theta)
```

**Arguments**

N	The size of the graph to generate
pi	A numeric vector of length K with clusters proportions (will be normalized to sum up to 1).
theta	A list of size V

**Details**

This function takes the desired graph size, cluster proportions and connectivity matrix as input and sample a graph accordingly together with the clusters labels.

**Value**

A list with fields:

- x: the multi-graph adjacency matrix as an array
- K: number of generated clusters
- N: number of vertex
- cl: vector of clusters labels
- pi: clusters proportions
- theta:

**Examples**

```
theta <- list(
  matrix(c(0.1, 0.9, 0.9, 0.1, 0.5, 0.5, 0.3, 0.7), ncol = 2, byrow = TRUE),
  matrix(c(0.5, 0.5, 0.3, 0.7, 0.05, 0.95, 0.3, 0.7), ncol = 2, byrow = TRUE),
  matrix(c(0.5, 0.5, 0.9, 0.1, 0.5, 0.5, 0.1, 0.9), ncol = 2, byrow = TRUE)
)
lca.data <- rlca(100, rep(1 / 4, 4), theta)
```

---

rmm

*Generate data using a Multinomial Mixture*


---

**Description**

rmm returns a count matrix and the cluster labels generated randomly with a Mixture of Multinomial model.

**Usage**

```
rmm(N, pi, mu, lambda)
```

**Arguments**

N	A numeric value the size of the graph to generate
pi	A numeric vector of length K with clusters proportions. Must sum up to 1.
mu	A numeric matrix of dim k x D with the clusters patterns to generate, all elements in [0,1].
lambda	A numeric value which specify the expectation for the row sums.

**Details**

It takes the sample size, cluster proportions and emission matrix, and as input and sample a graph accordingly together with the clusters labels.

**Value**

A list with fields:

- x: the count matrix as a dgCMatrix
- K: number of generated clusters
- N: number of vertex
- cl: vector of clusters labels
- pi: clusters proportions
- mu: connectivity matrix
- lambda: expectation of row sums

---

 rmreg

*Generate data from a mixture of regression model*


---

**Description**

rmreg returns an X matrix, a y vector and the cluster labels generated randomly with a Mixture of regression model.

**Usage**

```
rmreg(
  N,
  pi,
  A,
  sigma,
  X = cbind(rep(1, N), matrix(stats::rnorm(N * (ncol(A) - 1)), N, ncol(A) - 1))
)
```

**Arguments**

N	A numeric value the size of the graph to generate
pi	A numeric vector of length K with clusters proportions (must sum up to 1)
A	A numeric matrix of dim K x d with the regression coefficient
sigma	A numeric of length 1 with the target conditional variance
X	A matrix of covariate

**Details**

It takes the sample size, cluster proportions and regression parameters matrix and variance as input accordingly

**Value**

A list with fields:

- X: the covariate matrix
- y: the target feature
- K: number of generated clusters
- N: sample size
- cl: vector of clusters labels
- pi: clusters proportions
- A: regression coefficients used in the simulation
- sigma: conditional variance

---

rmultsbm

*Generate a graph adjacency matrix using a Stochastic Block Model*

---

**Description**

rmultsbm returns the multi-graph adjacency matrix and the cluster labels generated randomly with a Multinomial Stochastic Block Model.

**Usage**

```
rmultsbm(N, pi, mu, lambda)
```

**Arguments**

N	The size of the graph to generate
pi	A numeric vector of length K with clusters proportions (will be normalized to sum up to 1).
mu	A numeric array of dim K x K x M with the connectivity pattern to generate. elements in [0,1].
lambda	A double with the Poisson intensity to generate the total counts

**Details**

This function takes the desired graph size, cluster proportions and connectivity matrix as input and sample a graph accordingly together with the clusters labels.

**Value**

A list with fields:

- x: the multi-graph adjacency matrix as an array
- K: number of generated clusters
- N: number of vertex
- cl: vector of clusters labels
- pi: clusters proportions
- mu: connectivity matrix
- lambda:

**Examples**

```
simu <- rsbm(100, rep(1 / 5, 5), diag(rep(0.1, 5)) + 0.001)
```

---

rsbm

*Generate a graph adjacency matrix using a Stochastic Block Model*

---

**Description**

rsbm returns the adjacency matrix and the cluster labels generated randomly with a Stochastic Block Model.

**Usage**

```
rsbm(N, pi, mu)
```

**Arguments**

N	The size of the graph to generate
pi	A numeric vector of length K with clusters proportions (will be normalized to sum up to 1).
mu	A numeric matrix of dim K x K with the connectivity pattern to generate. elements in [0,1].

**Details**

This function takes the desired graph size, cluster proportions and connectivity matrix as input and sample a graph accordingly together with the clusters labels.



**Value**

A list with fields:

- x: the graph adjacency matrix as a dgCMatrix
- K: number of generated clusters
- N: number of vertex
- cl: vector of clusters labels
- pi: clusters proportions
- mu: connectivity matrix

**Examples**

```
simu <- rsbm(100, rep(1 / 5, 5), diag(rep(0.1, 5)) + 0.001)
```

---

Sbm

*Stochastic Block Model Prior class*

---

**Description**

An S4 class to represent a Stochastic Block Model. Such model can be used to cluster graph vertex, and model a square adjacency matrix  $X$  with the following generative model :

$$\pi \sim \text{Dirichlet}(\alpha)$$

$$Z_i \sim \mathcal{M}(1, \pi)$$

$$\theta_{kl} \sim \text{Beta}(a_0, b_0)$$

$$X_{ij} | Z_{ik} Z_{jl} = 1 \sim \mathcal{B}(\theta_{kl})$$

These classes mainly store the prior parameters value  $\alpha, a_0, b_0$  of this generative model. The Sbm-class must be used when fitting a simple Sbm whereas the SbmPrior-class must be used when fitting a [CombinedModels-class](#).

**Usage**

```
SbmPrior(a0 = 1, b0 = 1, type = "guess")
```

```
Sbm(alpha = 1, a0 = 1, b0 = 1, type = "guess")
```

**Arguments**

- |       |   |
|-------|---|
| a0    | Beta prior parameter over links (default to 1)  |
| b0    | Beta prior parameter over no-links (default to 1)   |
| type  | define the type of networks (either "directed", "undirected" or "guess", default to "guess"), for undirected graphs the adjacency matrix is supposed to be symmetric. |
| alpha | Dirichlet prior parameter over the cluster proportions (default to 1)   |

**Value**

a SbmPrior-class object  
 a Sbm-class object

**References**

Nowicki, Krzysztof and Tom A B Snijders (2001). “Estimation and prediction for stochastic block structures”. In: Journal of the American statistical association 96.455, pp. 1077–1087

**See Also**

[greed](#)

[SbmFit-class](#), [SbmPath-class](#)

Other DlvMModels: [CombinedModels](#), [DcLbm](#), [DcSbm](#), [DiagGmm](#), [DlvMPrior-class](#), [Gmm](#), [Lca](#), [MoM](#), [MoR](#), [MultSbm](#), [greed\(\)](#)

**Examples**

```
Sbm()
SbmPrior()
SbmPrior(type = "undirected")
Sbm()
Sbm(type = "undirected")
```

---

SbmFit-class

*Stochastic Block Model fit results class*

---

**Description**

An S4 class to represent a fit of a Stochastic Block Model, extend [IclFit-class](#).

**Slots**

`model` a [Sbm-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over rows and columns

`c1` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size `K` with number of elements in each clusters
- `x_counts`: matrix of size `K*K` with the number of links between each pair of clusters

`move_mat` binary matrix which store move constraints

`train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[coef, SbmFit-method](#)

---

SbmPath-class

*Stochastic Block Model hierarchical fit results class*

---

**Description**

An S4 class to represent a hierarchical fit of a stochastic block model, extend [IclPath-class](#).

**Slots**

`model` a [Sbm-class](#) object to store the model fitted

`name` generative model name

`icl` icl value of the fitted model

`K` number of extracted clusters over row and columns

`c1` a numeric vector with row and columns cluster indexes

`obs_stats` a list with the following elements:

- `counts`: numeric vector of size `K` with number of elements in each clusters
- `x_counts`: matrix of size `K*K` with the number of links between each pair of clusters

`path` a list of size `K-1` with that store all the solutions along the path. Each element is a list with the following fields:

- `icl1`: icl value reach with this solution for `alpha=1`
- `logalpha`: `log(alpha)` value were this solution is better than its parent
- `K`: number of clusters
- `c1`: vector of cluster indexes
- `k,l`: index of the cluster that were merged at this step
- `merge_mat`: lower triangular matrix of delta icl values
- `obs_stats`: a list with the following elements:
  - `counts`: numeric vector of size `K` with number of elements in each clusters
  - `x_counts`: matrix of size `K*K` with the number of links between each pair of clusters

`logalpha` value of `log(alpha)`

`ggtree` data.frame with complete merge tree for easy plotting with `ggplot2`

`tree` numeric vector with merge tree `tree[i]` contains the index of `i` father

`train_hist` data.frame with training history information (details depends on the training procedure)

**See Also**

[plot, SbmFit, missing-method](#)

Seed-class

*Greedy algorithm with seeded initialization***Description**

An S4 class to represent a greedy algorithm with initialization from spectral clustering and or k-means (extends [Alg-class](#) class ).

**Usage**

Seed()

**Value**

a Seed-class object

**Functions**

- Seed: Seed algorithm class constructor

**Examples**

Seed()

SevenGraders

*SevenGraders data***Description**

[SevenGraders](#) A small multiplex network of friendships among 29 seventh grade students in Victoria, Australia. Students nominated classmates for three different activities (who do you get on with in the class, who are your best friends, and who would you prefer to work with). Edge direction for each of these three types of edges indicates if node  $i$  nominated node  $j$ , and the edge weight gives the frequency of this nomination. Students 1-12 are boys and 13-29 are girls. The KONECT version of this network is the collapse of de Domenico's multiplex version.

**Usage**

data(SevenGraders)

**Format**

A binary array of size (29,29,3) containing directed graphs. The first layer encodes "getting along in class" while the second encodes the best-friendship (can be one-way). The third encodes the preferred work relation.

**Source**

[https://networks.skewed.de/net/7th\\_graders](https://networks.skewed.de/net/7th_graders)

**References**

M. Vickers and S. Chan, "Representing Classroom Social Structure." Melbourne: Victoria Institute of Secondary Education, (1981).

**Examples**

```
data(SevenGraders)
```

---

```
show,IclFit-method    Show an IclPath object
```

---

**Description**

Print an `IclPath-class` object, model type and number of found clusters are provided.

**Usage**

```
## S4 method for signature 'IclFit'
show(object)
```

**Arguments**

`object` `IclPath-class` object to print

**Value**

None (invisible NULL). No return value, called for side effects.

---

```
spectral    Regularized spectral clustering
```

---

**Description**

performs regularized spectral clustering of a sparse adjacency matrix

**Usage**

```
spectral(X, K)
```

**Arguments**

`X` An adjacency matrix in sparse format (see the `Matrix` package)  
`K` Desired number of cluster

**Value**

cl Vector of cluster labels

**References**

Tai Qin, Karl Rohe. Regularized Spectral Clustering under the Degree-Corrected Stochastic Block Model. Nips 2013.

---

to_multinomial	<i>Convert a binary adjacency matrix with missing value to a cube</i>
----------------	---

---

**Description**

Convert a binary adjacency matrix with missing value to a cube

**Usage**

```
to_multinomial(X)
```

**Arguments**

x                    A binary adjacency matrix with NA

**Value**

a cube

---

Youngpeoplesurvey	<i>Young People survey data</i>
-------------------	---------------------------------

---

**Description**

Young people survey data from Miroslav Sabo and available on the Kaggle platform. This is an authentic example of questionnaire data where Slovakian young people (15-30 years old) were asked musical preferences according to different genres (rock, hip-hop, classical, etc.).

**Usage**

```
data(Youngpeoplesurvey)
```

**Format**

An R data.frame with columns containing each of the 150 original variables of the study.

**Source**

<https://www.kaggle.com/miroslavsabo/young-people-survey>

**Examples**

```
data(Youngpeoplesurvey)
```

# Index

- \* **DlvmCoModels**
  - DlvmCoPrior-class, 25
- \* **DlvmModels**
  - CombinedModels, 12
  - DcLbm, 15
  - DcSbm, 18
  - DiagGmm, 22
  - DlvmPrior-class, 25
  - Gmm, 29
  - greed, 32
  - Lca, 37
  - MoM, 40
  - MoR, 43
  - MultSbm, 46
  - Sbm, 65
- \* **datasets**
  - Books, 5
  - fashion, 26
  - Fifa, 27
  - Football, 27
  - Jazz, 36
  - mushroom, 49
  - Ndrangheta, 49
  - NewGuinea, 50
  - SevenGraders, 68
  - Youngpeoplesurvey, 70
- Alg-class, 4
- available\_algorithms, 4
- available\_models, 4
- Books, 5
- clustering, 5
- clustering, IclFit-method (clustering), 5
- coef, DcLbmFit-method, 6
- coef, DcSbmFit-method, 7
- coef, DiagGmmFit-method, 8
- coef, GmmFit-method, 8
- coef, IclFit-method, 9
- coef, LcaFit-method, 9
- coef, MoMFit-method, 10
- coef, MoRFit-method, 10
- coef, MultSbmFit-method, 11
- coef, SbmFit-method, 11
- CombinedModels, 12, 16, 19, 23, 25, 30, 33, 38, 41, 44, 47, 66
- CombinedModels-class (CombinedModels), 12
- CombinedModelsFit-class, 13
- CombinedModelsPath-class, 13, 25
- cut, DcLbmPath-method, 14
- cut, IclPath-method, 15
- DcLbm, 12, 15, 19, 23, 25, 30, 33, 38, 41, 44, 47, 66
- DcLbm-class (DcLbm), 15
- DcLbmFit-class, 6, 16, 51
- DcLbmPath-class, 17, 52
- DcLbmPrior (DcLbm), 15
- DcLbmPrior-class (DcLbm), 15
- DcSbm, 12, 16, 18, 23, 25, 30, 33, 38, 41, 44, 47, 66
- DcSbm-class (DcSbm), 18
- DcSbmFit-class, 7, 20, 53
- DcSbmPath-class, 20
- DcSbmPrior (DcSbm), 18
- DcSbmPrior-class (DcSbm), 18
- DiagGmm, 12, 16, 19, 22, 25, 30, 33, 38, 41, 44, 47, 66
- DiagGmm-class (DiagGmm), 22
- DiagGmmFit-class, 8, 23, 53
- DiagGmmPath-class, 24
- DiagGmmPrior (DiagGmm), 22
- DiagGmmPrior-class (DiagGmm), 22
- DlvmCoPrior-class, 25
- DlvmPrior-class, 25
- extractSubModel, 25



- extractSubModel, CombinedModelsPath, character-member (extractSubModel), 25
- fashion, 26
- Fifa, 27
- Football, 27
- formula, 43
- Genetic (Genetic-class), 28
- Genetic-class, 28
- ggplot2, 31, 52–57
- Gmm, 12, 16, 19, 23, 25, 29, 32, 33, 38, 41, 44, 47, 66
- Gmm-class (Gmm), 29
- GmmFit-class, 8, 30, 54
- gmpairs, 31
- GmmPath-class, 31
- GmmPrior (Gmm), 29
- GmmPrior-class (Gmm), 29
- greed, 12, 16, 19, 23, 25, 26, 30, 32, 32, 38, 41, 44, 47, 66
- H, 33
- Hybrid (Hybrid-class), 34
- Hybrid-class, 34
- ICL, 35
- ICL, IclFit-method (ICL), 35
- IclFit-class, 5, 9, 35, 35, 37, 58
- IclPath-class, 36, 55
- Jazz, 36
- K, 37
- K, IclFit-method (K), 37
- Lca, 12, 16, 19, 23, 25, 30, 33, 37, 41, 44, 47, 66
- Lca-class (Lca), 37
- LcaFit-class, 9, 38, 55
- LcaPath-class, 39
- LcaPrior (Lca), 37
- LcaPrior-class (Lca), 37
- MI, 40
- MoM, 12, 16, 19, 23, 25, 30, 33, 38, 40, 44, 47, 66
- MoM-class (MoM), 40
- MoMFit-class, 10, 41, 56
- MoMPath-class, 42
- MoMPrior (MoM), 40
- MoMPrior-class (MoM), 40
- MoR, 12, 16, 19, 23, 25, 30, 33, 38, 41, 43, 47, 66
- MoR-class (MoR), 43
- MoRFit-class, 10, 44
- MoRPath-class, 45
- MoRPrior (MoR), 43
- MoRPrior-class (MoR), 43
- Multistarts (Multistarts-class), 46
- Multistarts-class, 46
- MultSbm, 12, 16, 19, 23, 25, 30, 33, 38, 41, 44, 46, 66
- MultSbm-class (MultSbm), 46
- MultSbmFit-class, 11, 47, 56
- MultSbmPath-class, 48
- MultSbmPrior (MultSbm), 46
- MultSbmPrior-class (MultSbm), 46
- mushroom, 49
- Ndrangheta, 49
- NewGuinea, 50, 50
- NMI, 51
- plot, DclbmFit, missing-method, 51
- plot, DclbmPath, missing-method, 52
- plot, DcSbmFit, missing-method, 53
- plot, DiagGmmFit, missing-method, 53
- plot, GmmFit, missing-method, 54
- plot, IclPath, missing-method, 55
- plot, LcaFit, missing-method, 55
- plot, MoMFit, missing-method, 56
- plot, MultSbmFit, missing-method, 56
- plot, SbmFit, missing-method, 57
- prior, 58
- prior, IclFit-method (prior), 58
- rdcsbm, 58
- rlbm, 59
- rlca, 60
- rmm, 61
- rmreg, 62
- rmultsbm, 63
- rsbm, 64
- Sbm, 12, 16, 19, 23, 25, 30, 32, 33, 38, 41, 44, 47, 65
- Sbm-class (Sbm), 65
- SbmFit-class, 11, 57, 66

SbmPath-class, [67](#)  
SbmPrior (Sbm), [65](#)  
SbmPrior-class (Sbm), [65](#)  
Seed (Seed-class), [68](#)  
Seed-class, [68](#)  
SevenGraders, [68](#), [68](#)  
show, IclFit-method, [69](#)  
sparseMatrix, [5](#), [27](#), [36](#)  
spectral, [69](#)  
  
to\_multinomial, [70](#)  
  
Youngpeoplesurvey, [70](#)