Package 'blockcluster'

July 22, 2025

Type Package

Title Co-Clustering Package for Binary, Categorical, Contingency and Continuous Data-Sets

Version 4.5.5

Encoding UTF-8

Date 2024-01-23

Copyright Inria

Description Simultaneous clustering of rows and columns, usually designated by biclustering, co-clustering or block clustering, is an important technique in two way data analysis. It consists of estimating a mixture model which takes into account the block clustering problem on both the individual and variables sets. The 'blockcluster' package provides a bridge between the C++ core library build on top of the 'STK++' library, and the R statistical computing environment. This package allows to co-cluster binary <doi:10.1016/j.csda.2007.09.007>, contingency <doi:10.1080/03610920903140197>, continuous <doi:10.1007/s11634-013-0161-3> and categorical data-sets <doi:10.1007/s11222-014-9472-2>. It also provides utility functions to visualize the results. This package may be useful for various applications in fields of Data mining, Information retrieval, Biology, computer vision and many more. More information about the project and comprehensive tutorial can be found on the link mentioned in URL.

License GPL (>= 3)

URL https://gitlab.inria.fr/iovleff/blockcluster

LazyLoad yes

Depends R (>= 4.1.0), rtkore (>= 1.6.10)

Imports methods

LinkingTo Rcpp, rtkore (>= 1.6.10)

SystemRequirements GNU make

Collate 'coclusterStrategy.R' 'RCocluster.R' 'optionclasses.R' 'cocluster.R' 'coclusterBinary.R' 'coclusterCategorical.R' 'coclusterContingency.R' 'coclusterContinuous.R' 'onattach.R'

NeedsCompilation yes

RoxygenNote 7.1.2

Author Serge Iovleff [aut, cre],
Parmeet Singh Bhatia [aut],
Josselin Demont [ctb],
Vincent Brault [ctb],
Vincent Kubicki [ctb],
Gerard Goavert [ctb],
Christophe Biernacki [ctb],
Gilles Celeux [ctb]

Maintainer Serge Iovleff <Serge.Iovleff@stkpp.org>

Repository CRAN

Date/Publication 2024-02-23 13:40:02 UTC

Contents

binarydata *Simulated Binary Data-set*

Description

It is a binary data-set simulated using Bernoulli distribution. It consist of two clusters in rows and three clusters in columns.

Format

A data matrix with 1000 rows and 100 columns.

Examples

```
data(binarydata)
```

BinaryOptions-class *Binary input/output options*

Description

This class contains all the input options as well as the estimated paramters for Binary data-set. It inherits from base class CommonOptions. The class contains following output parameters given in 'Details' along with the parameters in base class.

Details

classmean: The mean value of each co-cluster.

classdispersion: The dispersion of each co-cluster.

ICLvalue: Integrated complete likelihood

blockcluster *Co-Clustering Package*

Description

This package performs Co-clustering of binary, contingency, continuous and categorical data-sets.

Details

This package performs Co-clustering of binary, contingency, continuous and categorical data-sets with utility functions to visualize the Co-clustered data. The package contains a set of functions coclusterBinary, coclusterCategorical, coclusterContingency, coclusterContinuous which perform Co-clustering on various kinds of data-sets and return object of appropriate class (refer to documentation of these functions. The package also contains function coclusterStrategy (see documentation of function to know various slots) which returns an object of class strategy. This object can be given as input to co-clustering functions to control various Co-clustering parameters. Please refer to testmodels.R file which is included in "test" directory to see examples with various models and simulated data-sets.

The package also provide utility functions like summary() and plot() to summarize results and plot the original and Co-clustered data respectively.

Examples

```
## Simple example with simulated binary data
## load data
data(binarydata)
## usage of coclusterBinary function in its most simplest form
out<-coclusterBinary(binarydata,nbcocluster=c(2,3))
#" Summarize the output results
summary(out)
## Plot the original and Co-clustered data
plot(out)
```

---

categoricaldata *Simulated categorical Data-set*

Description

It is a categorical data-set simulated using Categorical distribution with 5 modalities. It consist of three clusters in rows and two clusters in columns.

Format

A data matrix with 1000 rows and 100 columns.

Examples

```
data(categoricaldata)
```

CategoricalOptions-class *Categorical input/output options*

Description

This class contains all the input options as well as the estimated paramters for categorical data-set. It inherits from base class CommonOptions. The class contains following output parameters given in 'Details' along with the parameters in base class.

Details

classmean: The categorical distribution of each co-cluster

ICLvalue: Integrated complete likelihood

cocluster *Co-Clustering function.*

Description

This function performs Co-Clustering (simultaneous clustering of rows and columns) for Binary, Contingency and Continuous data-sets using latent block models.It can also be used to perform semi-supervised co-clustering.

Usage

```
cocluster(
  data,
  datatype,
  semisupervised = FALSE,
  rowlabels = integer(0),
  collabels = integer(0),
  model = NULL,
  nbcocluster,
  strategy = coclusterStrategy(),
  nbCore = 1
)
```

Arguments

data	Input data as matrix (or list containing data matrix, numeric vector for row effects and numeric vector column effects in case of contingency data with known row and column effects.)
datatype	This is the type of data which can be "binary" , "contingency", "continuous" or "categorical".

`semisupervised` Boolean value specifying whether to perform semi-supervised co-clustering or not. Make sure to provide row and/or column labels if specified value is true. The default value is false.

`rowlabels` Integer Vector specifying the class of rows. The class number starts from zero. Provide -1 for unknown row class.

`collabels` Integer Vector specifying the class of columns. The class number starts from zero. Provide -1 for unknown column class.

`model` This is the name of model. The following models exists for various types of data:

Model	Data-type	Proportions	Dispersion/Variance
pik_rhol_epsilonkl(Default)	binary	unequal	unequal
pik_rhol_epsilon	binary	unequal	equal
pi_rho_epsilonkl	binary	equal	unequal
pi_rho_epsilon	binary	equal	equal
pik_rhol_sigma2kl(Default)	continuous	unequal	unequal
pik_rhol_sigma	continuous	unequal	equal
pi_rho_sigma2kl	continuous	equal	unequal
pi_rho_sigma2	continuous	equal	equal
pik_rhol_unknown(default)	contingency	unequal	N.A
pi_rho_unknown	contingency	equal	N.A
pik_rhol_known	contingency	unequal	N.A
pi_rho_known	contingency	equal	N.A
pik_rhol_multi	categorical	unequal	unequal
pi_rho_multi	categorical	equal	unequal

`nbcocluster` Integer vector specifying the number of row and column clusters respectively.

`strategy` Object of class `strategy`.

`nbCore` number of thread to use (OpenMP must be available), 0 for all cores. Default value is 1.

Value

Return an object of `BinaryOptions` or `ContingencyOptions` or `ContinuousOptions` depending on whether the data-type is Binary, Contingency or Continuous respectively.

Examples

```
# Simple example with simulated binary data
#load data
data(binarydata)
#usage of cocluster function in its most simplest form
out<-cocluster(binarydata,datatype="binary",nbcocluster=c(2,3))
#Summarize the output results
summary(out)
#Plot the original and Co-clustered data
plot(out)
```

coclusterBinary *Co-Clustering function for Binary data.*

Description

This function performs Co-Clustering (simultaneous clustering of rows and columns) for Binary data-sets using latent block models. It can also be used to perform semi-supervised co-clustering.

Usage

```
coclusterBinary(
  data,
  semisupervised = FALSE,
  rowlabels = integer(0),
  collabels = integer(0),
  model = NULL,
  nbcocluster,
  strategy = coclusterStrategy(),
  a = 1,
  b = 1,
  nbCore = 1
)
```

Arguments

data	Input data as matrix (or list containing data matrix)
semisupervised	Boolean value specifying whether to perform semi-supervised co-clustering or not. Make sure to provide row and/or column labels if specified value is true. The default value is false.
rowlabels	Integer Vector specifying the class of rows. The class number starts from zero. Provide -1 for unknown row class.
collabels	Integer Vector specifying the class of columns. The class number starts from zero. Provide -1 for unknown column class.
model	This is the name of model. The following models exists for Binary data:

Model	Data-type	Proportions	Dispersion/Variance
pik_rhol_epsilonkl(Default)	binary	unequal	unequal
pik_rhol_epsilon	binary	unequal	equal
pi_rho_epsilonkl	binary	equal	unequal
pi_rho_epsilon	binary	equal	equal

nbcocluster	Integer vector specifying the number of row and column clusters respectively.
strategy	Object of class strategy.
a	First hyper-parameter in case of Bayesian settings. Default is 1 (no prior).

b	Second hyper-parameter in case of Bayesian settings. Default is 1 (no prior).
nbCore	number of thread to use (OpenMP must be available), 0 for all cores. Default value is 1.

Value

Return an object of BinaryOptions.

Examples

```
## Simple example with simulated binary data
## load data
data(binarydata)
## usage of coclusterBinary function in its most simplest form
out<-coclusterBinary(binarydata,nbcocluster=c(2,3))
## Summarize the output results
summary(out)
## Plot the original and Co-clustered data
plot(out)
```

---

coclusterCategorical *Co-Clustering function for categorical data-sets.*

Description

This function performs Co-Clustering (simultaneous clustering of rows and columns) Categorical data-sets using latent block models. It can also be used to perform semi-supervised co-clustering.

Usage

```
coclusterCategorical(
  data,
  semisupervised = FALSE,
  rowlabels = integer(0),
  collabels = integer(0),
  model = NULL,
  nbcocluster,
  strategy = coclusterStrategy(),
  a = 1,
  b = 1,
  nbCore = 1
)
```

Arguments

`data`	Input data as matrix (or list containing data matrix.)
`semisupervised`	Boolean value specifying whether to perform semi-supervised co-clustering or not. Make sure to provide row and/or column labels if specified value is true. The default value is false.
`rowlabels`	Integer Vector specifying the class of rows. The class number starts from zero. Provide -1 for unknown row class.
`collabels`	Integer Vector specifying the class of columns. The class number starts from zero. Provide -1 for unknown column class.
`model`	This is the name of model. The following models exists for categorical data:

pik_rhol_multi	categorical	unequal	unequal
pi_rho_multi	categorical	equal	unequal

`nbcocluster`	Integer vector specifying the number of row and column clusters respectively.
`strategy`	Object of class `strategy`.
`a`	First hyper-parameter in case of Bayesian settings. Default is 1 (no prior).
`b`	Second hyper-parameter in case of Bayesian settings. Default is 1 (no prior).
`nbCore`	number of thread to use (OpenMP must be available), 0 for all cores. Default value is 1.

Value

Return an object of `BinaryOptions` or `ContingencyOptions` or `ContinuousOptions` depending on whether the data-type is Binary, Contingency or Continuous respectively.

Examples

```
## Simple example with simulated categorical data
## load data
data(categoricaldata)
## usage of coclusterCategorical function in its most simplest form
out<-coclusterCategorical(categoricaldata,nbcocluster=c(3,2))
## Summarize the output results
summary(out)
## Plot the original and Co-clustered data
plot(out)
```

`coclusterContingency` *Co-Clustering function.*

Description

This function performs Co-Clustering (simultaneous clustering of rows and columns) for Contingency data-sets using latent block models.It can also be used to perform semi-supervised co-clustering.

Usage

```
coclusterContingency(
  data,
  semisupervised = FALSE,
  rowlabels = integer(0),
  collabels = integer(0),
  model = NULL,
  nbcocluster,
  strategy = coclusterStrategy(),
  nbCore = 1
)
```

Arguments

`data`	Input data as matrix (or list containing data matrix, numeric vector for row effects and numeric vector column effects in case of contingency data with known row and column effects.)
`semisupervised`	Boolean value specifying whether to perform semi-supervised co-clustering or not. Make sure to provide row and/or column labels if specified value is true. The default value is false.
`rowlabels`	Integer Vector specifying the class of rows. The class number starts from zero. Provide -1 for unknown row class.
`collabels`	Integer Vector specifying the class of columns. The class number starts from zero. Provide -1 for unknown column class.
`model`	This is the name of model. The following models exists for Poisson data:

pik_rhol_unknown(default)	contingency	unequal	N.A
pi_rho_unknown	contingency	equal	N.A
pik_rhol_known	contingency	unequal	N.A
pi_rho_known	contingency	equal	N.A

`nbcocluster`	Integer vector specifying the number of row and column clusters respectively.
`strategy`	Object of class `strategy`.
`nbCore`	number of thread to use (OpenMP must be available), 0 for all cores. Default value is 1.

Value

Return an object of BinaryOptions or ContingencyOptions or ContinuousOptions depending on whether the data-type is Binary, Contingency or Continuous respectively.

Examples

```
## Simple example with simulated contingency data
## load data
data(contingencydataunknown)
## usage of coclusterContingency function in its most simplest form
strategy = coclusterStrategy( nbinititerations = 5, nbxem = 2, nbiterations_int = 2
                         , nbiterationsxem = 10, nbiterationsXEM = 100, epsilonXEM=1e-5)
out<-coclusterContingency( contingencydataunknown, nbcocluster=c(2,3), strategy = strategy)
## Summarize the output results
summary(out)
## Plot the original and Co-clustered data
plot(out)
```

`coclusterContinuous` *Co-Clustering function.*

Description

This function performs Co-Clustering (simultaneous clustering of rows and columns) for continuous data-sets using latent block models. It can also be used to perform semi-supervised co-clustering.

Usage

```
coclusterContinuous(
  data,
  semisupervised = FALSE,
  rowlabels = integer(0),
  collabels = integer(0),
  model = NULL,
  nbcocluster,
  strategy = coclusterStrategy(),
  nbCore = 1
)
```

Arguments

`data`	Input data as matrix (or list containing data matrix.)
`semisupervised`	Boolean value specifying whether to perform semi-supervised co-clustering or not. Make sure to provide row and/or column labels if specified value is true. The default value is false.

`rowlabels` Vector specifying the class of rows. The class number starts from zero. Provide -1 for unknown row class.

`collabels` Vector specifying the class of columns. The class number starts from zero. Provide -1 for unknown column class.

`model` This is the name of model. The following models exists for Gaussian data:

Model	Data-type	Proportions	Dispersion/Variance
pik_rhol_sigma2kl(Default)	continuous	unequal	unequal
pik_rhol_sigma2	continuous	unequal	equal
pi_rho_sigma2kl	continuous	equal	unequal
pi_rho_sigma2	continuous	equal	equal

`nbcocluster` Integer vector specifying the number of row and column clusters respectively.

`strategy` Object of class `strategy`.

`nbCore` number of thread to use (OpenMP must be available), 0 for all cores. Default value is 1.

Value

Return an object of `BinaryOptions` or `ContingencyOptions` or `ContinuousOptions` depending on whether the data-type is Binary, Contingency or Continuous respectively.

Examples

```
# Simple example with simulated continuous data
#load data
data(gaussiandata)
#usage of coclusterContinuous function in its most simplest form
out<-coclusterContinuous(gaussiandata,nbcocluster=c(2,3))
#Summarize the output results
summary(out)
#Plot the original and Co-clustered data
plot(out)
```

`coclusterStrategy` *Strategy function*

Description

This function is used to set all the parameters for Co-clustering. It returns an object of class `strategy` which can be given as input to `coclusterBinary`, `coclusterCategorical`, `coclusterContingency`, `coclusterContinuous` function.

This class contains all the input parameters to run coclustering.

Usage

```
coclusterStrategy(
  algo = "BEM",
  initmethod = "emInitStep",
  stopcriteria = "Parameter",
  nbiterationsxem = 50,
  nbiterationsXEM = 500,
  nbinitmax = 100,
  nbinititerations = 10,
  initepsilon = 0.01,
  nbiterations_int = 5,
  epsilon_int = 0.01,
  epsilonxem = 1e-04,
  epsilonXEM = 1e-10,
  nbtry = 2,
  nbxem = 5
)
```

Arguments

algo	The valid values for this parameter are "BEM" (Default), "BCEM", "BSEM" and "BGibbs" (only for Binary model).
initmethod	Method to initialize model parameters. The valid values are "cemInitStep", "emInitStep" and "randomInit".
stopcriteria	It specifies the stopping criteria. It can be based on either relative change in parameters (preffered due to computation reasons) value or relative change in pseudo log-likelihood. Valid criterion values are "Parameter" and "Likelihood". Default criteria is "Parameter".
nbiterationsxem	Number of EM iterations used during xem step. Default value is 50.
nbiterationsXEM	Number of EM iterations used during XEM step. Default value is 500.
nbinitmax	Maximal number initialization to try. Default value is 100.
nbinititerations	Number of Global iterations used in initialization step. Default value is 10.
initepsilon	Tolerance value used while initialization. Default value is 1e-2.
nbiterations_int	Number of iterations for internal E step. Default value is 5.
epsilon_int	Tolerance value for relative change in Parameter/likelihood for internal E-step. Default value is 1e-2.
epsilonxem	Tolerance value used during xem step. Default value is 1e-4.
epsilonXEM	Tolerance value used during XEM step. Default value is 1e-10
nbtry	Number of tries (XEM steps). Default value is 2.
nbxem	Number of xem steps. Default value is 5.

Details

algo: Algorithm to be use for co-clustering.

stopcriteria: Stopping criteria used to stop the algorithm.

initmethod: Method to initialize model parameters.

nbinitmax: Maximal number of initialization to try (if reached estimation failed)

nbinititerations: Number of global iterations while running initialization.

initepsilon: Tolerance value used while initialization.

nbiterations_int: Number of iterations for internal E-step.

epsilon_int: Tolerance value for internal E-step.

nbtry: Number of tries.

nbxem: Number of xem iterations.

nbiterationsxem: Number of EM iterations used during xem.

nbiterationsXEM: Number of EM iterations used during XEM.

epsilonxem: Tolerance value used during xem.

epsilonXEM: Tolerance value used during XEM.

Value

Object of class `strategy`

Examples

```
#Default strategy values

strategy<-coclusterStrategy()
summary(strategy)
```

`CommonOptions-class` *Common Input/Output options.*

Description

This class contains all the input options and common output options for all kinds of data-sets (Binary, Categorical, Contingency and Continuous).

Details

The following are the various input options:

data: Input data.

datatype: Type of data.

semisupervised: Boolean value specifying if Co-clustering is semi-supervised or not.

model: Model to be run for co-clustering.

nbcocluster: Number of row and column clusters.

strategy: Input strategy.

The following are the various common output options:

message: Status returned.

rowproportions: Vector of row proportions.

colproportions: Vector of column proportions.

rowclass: Vector of assigned row cluster to each row.

colclass: Vector of assigned column cluster to each column.

likelihood: Final pseudo log-likelihood.

rowposteriorprob: Final posterior probabilities for rows.

colposteriorprob: Final posterior probabilities for columns.

---

contingencydatalist *Simulated Contingency Data-set*

Description

It is a contingency data-set simulated using Poisson distribution. The row and column effects is unknown for this data-set. It consist of two clusters in rows and three clusters in columns.

Format

A data list consisting of following data:

data A data matrix consisting of 1000 rows and 100 columns.

roweffects A numeric vector of size 1000. Each value represent row effect of corresponding row.

columneffects A numeric vector of size 100. Each value represent column effect of corresponding column.

Examples

```
data(contingencydatalist)
```

`contingencydataunknown` *Simulated Contingency Data-set*

Description

It is a contingency data-set simulated using Poisson distribution. The row and column effects is unknown for this data-set. It consist of two clusters in rows and three clusters in columns.

Format

A data matrix with 1000 rows and 100 columns.

Examples

```
data(contingencydataunknown)
```

`ContingencyOptions-class` *Contingency input/output options*

Description

This class contains all the input options as well as the estimated paramters for Contingency data-set.It inherits from base class `CommonOptions`. The class contains following output parameters given in 'Details' along with the parameters in base class.

Details

classgamma: The value of poisson parameter (gamma) for each co-cluster.

datamui: Rows effect (if known).

datanuj: Columns effect (if known).

ContinuousOptions-class *Continuous input/output options*

Description

This class contains all the input options as well as the estimated parameters for Continuous data-sets. It inherits from base class CommonOptions. The class contains following output parameters given in 'Details' along with the parameters in base class.

Details

classmean: The mean value of each co-cluster.

classvariance: The variance of each co-cluster.

gaussiandata *Simulated Gaussian Data-set*

Description

It is a Continuous data-set simulated using Gaussian distribution. It consist of two clusters in rows and three clusters in columns.

Format

A data matrix with 1000 rows and 100 columns.

Examples

```
data(gaussiandata)
```

plot *Plot function.*

Description

This function plot the original and Co-clustered data-sets.

Usage

```
## S4 method for signature 'BinaryOptions'
plot(x, y, ...)

## S4 method for signature 'ContingencyOptions'
plot(x, y, ...)

## S4 method for signature 'ContinuousOptions'
plot(x, y, ...)

## S4 method for signature 'CategoricalOptions'
plot(x, y, ...)
```

Arguments

x	output object from coclusterBinary, coclusterCategorical, coclusterContingency, coclusterContinuous.
y	Ignored
...	Additional argument(s). Currently we support two additional argument. "asp": If this is set to TRUE the original aspect ratio is conserved. By default "asp" is FALSE. "type" : This is the type of plot which is either "cocluster" or "distribution". The corresponding plots are Co-clustered data and distributions and mixture densities for Co-clusters respectively. Default is "cocluster" plot.

---

summary,strategy-method

Summary function.

Description

This function gives the summary of output from coclusterBinary, coclusterCategorical, coclusterContingency, coclusterContinuous.

Usage

```
## S4 method for signature 'strategy'
summary(object, ...)

## S4 method for signature 'BinaryOptions'
summary(object, ...)

## S4 method for signature 'ContingencyOptions'
summary(object, ...)

## S4 method for signature 'ContinuousOptions'
summary(object, ...)
```

```
## S4 method for signature 'CategoricalOptions'
summary(object, ...)
```

Arguments

object	output object from coclusterBinary, coclusterCategorical, coclusterContingency, coclusterContinuous.
...	Additional argument(s) . Currently there is no additional arguments.

XEMStrategy *An EM strategy to obtain a good optimum.*

Description

In Co-clustering, there could be many local optimal where the algorithm may get struck resulting in sub-optimum results. Hence we applied a strategy called XEM strategy to run the EM algorithm. The various steps are defined as follows:

Details

Step-1, "xem" step: Do several runs of: "initialization followed by short run of algorithm (few iterations/high tolerance)". This parameter is named as "nbxem" in coclusterStrategy function. Default value is 5. We call this step as xem step.

Step-2, "XEM" step: Select the best result of step 1 and make long run of Algorithm(high iterations/low tolerance).We call this step as XEM step.

Step-3 Repeat step 1 and 2 several times and select the best result. The number of repetitions can be modified via parameter "nbtry" of coclusterStrategy function. Default value is 2.

[,strategy-method *Getter method for blockcluster output*

Description

This is overloading of square braces to extract values of various slots of the output from coclusterBinary, coclusterCategorical, coclusterContingency, coclusterContinuous.

Usage

```
## S4 method for signature 'strategy'
x[i, j, drop]

## S4 method for signature 'BinaryOptions'
x[i, j, drop]

## S4 method for signature 'ContingencyOptions'
x[i, j, drop]

## S4 method for signature 'ContinuousOptions'
x[i, j, drop]

## S4 method for signature 'CategoricalOptions'
x[i, j, drop]
```

Arguments

x	object from which to extract element(s) or in which to replace element(s).
i	the name of the element we want to extract or replace.
j	if the element designing by i is complex, j specifying elements to extract or replace.
drop	not used

Index