Package ‘ordinalClust’

Type Package
Title Ordinal Data Clustering, Co-Clustering and Classification
Version 1.3.4
Date 2019-07-25
Author Margot Selosse, Julien Jacques, Christophe Biernacki
Maintainer Margot Selosse <margot.selosse@gmail.com>
License GPL (>= 2)
Imports Rcpp (>= 0.12.11), methods
LinkingTo Rcpp, RcppProgress, RcppArmadillo, BH
Suggests knitr, rmarkdown, caret, ggplot2
VignetteBuilder knitr
LazyData true
SystemRequirements C++11
Depends R (>= 3.3)
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-07-25 12:40:02 UTC

R topics documented:

boscoclust .................................. 4
bosclust .................................. 6
boscoclust .................................. 6
dataqol .................................. 9
dataqol.classif .................................. 9
Msimulated .................................. 10
pejSim .................................. 10
bosclassif

Function to perform a classification

Description

This function performs a classification on a dataset with features of the ordinal kind, and a label variable of the integer type (1, 2, ..., kr). The classification function proposes two classification models. The first one, (chosen by the argument kc=0), is a multivariate BOS model assuming that, conditionally on the class of the observations, the features are independent. The second model is a parsimonious version of the first model. Parsimony is introduced by grouping the features into clusters (as in co-clustering) and assuming that the features of a cluster have a common distribution. If the data contains ordinal features with D different numbers of levels, the data is going to be seen as D matrices of ordinal data.

Usage

    bosclassif(x, y, idx_list=c(1), kr, kc=0, init, nbSEM, nbSEMburn,
               nbbindmini, m=0, percentRandomB=0)

Arguments

x       Matrix made of ordinal data, of dimension N*Jtot. The features with same numbers of levels must be placed side by side. The missing values should be coded as NA.

y       Vector of length N. It should represent the classes corresponding to each row of x. Must be labelled with integers (1, 2, ..., kr).

idx_list       Vector of length D. This argument is useful when variables have different numbers of levels. Element d should indicate where the variables with number of levels m[d] begins in matrix x.

kr       Number of row classes.

kc       Vector of length D. d^th element indicates the number of column clusters. Set to 0 to choose a classical multivariate BOS model.

m       Vector of length D. d^th element defines the ordinal data’s number of levels.

nbSEM       Number of SEM-Gibbs iterations realized to estimate parameters.

nbSEMburn       Number of SEM-Gibbs burning iterations for estimating parameters. This parameter must be inferior to nbSEM.

nbbindmini       Minimum number of cells belonging to a block.

init       String that indicates the kind of initialisation. Must be one of the following words: "kmeans", "random" or "randomBurnin".

percentRandomB       Vector of length 1. Indicates the percentage of resampling when init is equal to "randomBurnin".
**Value**

Return an object. The slots are:

- **@zr** Vector of length N with resulting row partitions.
- **@zc** List of length D. d\(^{th}\) item is a vector of length J[d] representing the columns partitions for the group of variables d.
- **@J** Vector of length D. d\(^{th}\) item represents the number of columns for d\(^{th}\) group of variables.
- **@W** List of length D. Item d is a matrix of dimension J*kc[d] such that W[j,h]=1 if j belongs to cluster h.
- **@V** Matrix of dimension N*kr such that V[i,g]=1 if i belongs to cluster g.
- **@icl** ICL value for co-clustering.
- **@kr** Number of row classes.
- **@name** Name of the result.
- **@number_distrib** Number of groups of variables.
- **@pi** Vector of length kr. Row mixing proportions.
- **@rho** List of length D. d\(^{th}\) item represents the column mixing proportion for d\(^{th}\) group of variables.
- **@dlist** List of length d. d\(^{th}\) item represents the indexes of group of variables d.
- **@kc** Vector of length D. d\(^{th}\) element represents the number of clusters column H for d\(^{th}\) group of variables.
- **@m** Vector of length D. d\(^{th}\) element represents the number of levels of d\(^{th}\) group of variables.
- **@nbSEM** Number of SEM-Gibbs algorithm iteration.
- **@params** List of length D. d\(^{th}\) item represents the blocks parameters for group of variables d.
- **@xhat** List of length D. d\(^{th}\) item represents the d\(^{th}\) group of variables dataset, with missing values completed.

**Author(s)**

Margot Selosse, Julien Jacques, Christophe Biernacki.

**Examples**

```r
# loading the real dataset
data("dataqol.classif")
set.seed(5)

# loading the ordinal data
M <- as.matrix(dataqol.classif[,2:29])
```
# creating the classes values
y <- as.vector(dataqol.classif$death)

# sampling datasets for training and to predict
nb.sample <- ceiling(nrow(M)*2/3)
sample.train <- sample(1:nrow(M), nb.sample, replace=FALSE)
M.train <- M[sample.train,]
M.validation <- M[-sample.train,]
nb.missing.validation <- length(which(M.validation==0))
m <- c(4)
M.validation[which(M.validation==0)] <- sample(1:m, nb.missing.validation,replace=TRUE)

y.train <- y[sample.train]
y.validation <- y[-sample.train]

# configuration for SEM algorithm
nbSEM=50
nbSEMburn=40
nbindmini=1
init="kmeans"

# number of classes to predict
kr <- 2
# different kc to test with cross-validation
kcol <- 1

res <- bosclassif(x=M.train,y=y.train,kr=kr,kc=kcol,m=m,
                   nbSEM=nbSEM,nbSEMburn=nbSEMburn,
                   nbindmini=nbindmini,init=init)
predictions <- predict(res, M.validation)

---

bosclust  
*Function to perform a clustering*

**Description**

This function performs a clustering on ordinal data by using the multiple latent block model (cf references for further details). It allows the user to define D groups of variables that have different number of levels. A BOS distribution is used, and the parameters inference is realized with an SEM-Gbbs algorithm.
bosclust

Usage

bosclust(x, idx_list=c(1), kr, init, nbSEM, nbSEMburn, nbindmini, m=0, percentRandomB=0)

Arguments

x
Matrix made of ordinal data, of dimension N*Jtot. The features with same numbers of levels must be placed side by side. The missing values should be coded as NA.

idx_list
Vector of length D. This argument is useful when variables have different numbers of levels. Element d should indicate where the variables with number of levels m[d] begins in matrix x.

kr
Number of row clusters.

m
Vector of length D. d^th element defines the ordinal data’s number of levels.

nbSEM
Number of SEM-Gibbs iterations realized to estimate parameters.

nbSEMburn
Number of SEM-Gibbs burning iterations for estimating parameters. This parameter must be inferior to nbSEM.

nbindmini
Minimum number of cells belonging to a block.

init
String that indicates the kind of initialisation. Must be one of th following words : "kmeans", "random" or "randomBurnin".

percentRandomB
Vector of length 1. Indicates the percentage of resampling when init is equal to "randomBurnin".

Value

@V
Matrix of dimension N*kr such that V[i,g]=1 if i belongs to cluster g.

@zr
Vector of length N with resulting row partitions.

@pi
Vector of length kr. Row mixing proportions.

@m
Vector of length D. d^th element represents the number of levels of d^th group of variables.

@icl
ICL value for clustering.

@name
Name of the result.

@params
List of length D. d^th item stores the resulting position and precision parameters mu and pi.

@paramschain
List of length nbSEMburn. For each iteration of the SEM-Gibbs algorithm, the parameters of the blocks are stored.

@xhat
List of length D. d^th item represents the d^th group of variables dataset, with missing values completed.

@zrchain
Matrix of dimension nbSEM*N. Row i represents the row cluster partitions at iteration i.

@pichain
List of length nbSEM. Item i is a vector of length kr which contains the row mixing proportions at iteration i.
**boscoclust**

*Function to perform a co-clustering*

**Description**

This function performs a co-clustering on ordinal data by using the latent block model (cf references for further details). A BOS distribution is used, and the parameters inference is realized with an SEM-Gbbs algorithm.

**Usage**

```r
boscoclust(x = matrix(0, nrow = 1, ncol = 1), idx_list = c(1), kr, kc, init, nbSEM, nbSEMburn, nbRepeat = 1, nbindmini, m = 0, percentRandomB = 0)
```

**Arguments**

- `x` Matrix made of ordinal data, of dimension N*Jtot. The features with same numbers of levels must be placed side by side. The missing values should be coded as NA.
idx_list Vector of length D. This argument is useful when variables have different numbers of levels. Element d should indicate where the variables with number of levels m[d] begins in matrix x.

kr Number of row classes.

kc Vector of length D. d^th element indicates the number of column clusters. Set to 0 to choose a classical multivariate BOS model.

m Vector of length D. d^th element defines the ordinal data’s number of levels.

nbSEM Number of SEM-Gibbs iterations realized to estimate parameters.

nbSEMBurn Number of SEM-Gibbs burning iterations for estimating parameters. This parameter must be inferior to nbSEM.

nbRepeat Number of times sampling on rows and on columns will be done at each SEM-Gibbs iteration.

nbmin Vector Minimum number of cells belonging to a block.

init String that indicates the kind of initialisation. Must be one of the following words: "kmeans", "random" or "randomBurnin".

percentRandomB Vector of length 2. Indicates the percentage of resampling when init is equal to "randomBurnin".

Value

@V Matrix of dimension N*kr such that V[i,g]=1 if i belongs to cluster g.
@icl ICL value for co-clustering.
@name @paramschain List of length nbSEMBurn. For each iteration of the SEM-Gibbs algorithm, the parameters of the blocks are stored.
@pichain List of length nbSEM. Item i is a vector of length kr which contains the row mixing proportions at iteration i.
@rhochain List of length nbSEM. Item i is a list of length D whose d^th contains the column mixing proportions of groups of variables d, at iteration i.
@zc List of length D. d^th item is a vector of length J[d] representing the columns partitions for the group of variables d.
@zr Vector of length N with resulting row partitions.
@W List of length D. Item d is a matrix of dimension J*kc[d] such that W[j,h]=1 if j belongs to cluster h.
@m Vector of length D. d^th element represents the number of levels of d^th group of variables.
@params List of length D. d^th item represents the blocks parameters for group of variables d.
@pi Vector of length kr. Row mixing proportions.
@rho List of length D. d^th item represents the column mixing proportion for d^th group of variables.
@xhat List of length D. d\textsuperscript{th} item represents the d\textsuperscript{th} group of variables dataset, with missing values completed.

@zrchain Matrix of dimension nbSEM*\(N\). Row i represents the row cluster partitions at iteration i.

@zrchain List of length D. Item d is a matrix of dimension nbSEM*J[d]. Row i represents the column cluster partitions at iteration i.

Author(s)

Margot Selosse, Julien Jacques, Christophe Biernacki.

Examples

```r
library(ordinalClust)

# loading the real dataset
data("dataqol")
set.seed(5)

# loading the ordinal data
M <- as.matrix(dataqol[,2:29])

# defining different number of categories:
m=4

# defining number of row and column clusters
krow = 5
kcol = 4

# configuration for the inference
nbSEM=50
nbSEMburn=40
nbindmini=2
init = "kmeans"

# Co-clustering execution
object <- boscoclust(x=M, kr=krow, kc=kcol, m=m, nbSEM=nbSEM, nbSEMburn=nbSEMburn, nbindmini=nbindmini, init=init)
```
**dataqol**

*Questionnaires Responses Of Patients Affected By Breast Cancer*

**Description**

This dataset contains the responses of 121 patients to 30 questions about their quality of life.

**Usage**

`dataqol`

**Format**

A dataframe with 121 lines and 31 columns. A line represents a patient and a column are information about the patients

- **Id** patient Id
- **q1-q28** responses to 28 questions with number of levels equals to 4
- **q29-q30** responses to 22 questions with number of levels equals to 7

**Source**

The table was determined based on data associated with the package available on: https://cran.r-project.org/package=QoLR

---

**dataqol.classif**

*Questionnaires Responses Of Patients Affected By Breast Cancer*

**Description**

This dataset contains the responses of 40 patients to 30 questions about their quality of life. Furthermore, a variable indicates if the patient survived from the disease.

**Usage**

`dataqol.classif`

**Format**

A dataframe with 40 lines and 32 columns. A line represents a patient and a column are information about the patients

- **Id** patient Id
- **q1-q28** responses to 28 questions with number of levels equals to 4
- **q29-q30** responses to 22 questions with number of levels equals to 7
- **death** if the patient survived (1) or not (2)
**Source**

The table was determined based on data associated with the package available on: [https://cran.r-project.org/package=QoLR](https://cran.r-project.org/package=QoLR)

---

**Msimulated**

*Matrix of simulated ordinal data*

---

**Description**

This is a toy dataset for running simple examples.

**Usage**

Msimulated

---

**Format**

An ordinal data matrix with 60 lines and 50 columns. Number of levels is equal to 3. 4 blocks are simulated with (μ,π) parameters equal to (3,0.5), (2,0.7), (1,0.8) and (2,0.6).

---

**pejSim**

*pejSim*

---

**Description**

This function computes the probability for a level ej to be sampled from a BOS distribution of parameters (μ,π), with a number of levels equals to m. Can be used to generate data with BOS distribution.

**Usage**

pejSim(ej,m,μ,π)

---

**Arguments**

- **ej** levels to be sampled
- **m** Number of levels.
- **μ** μ parameter for BOS distribution.
- **π** π parameter for BOS distribution.

**Value**

Return the probability of ej to be sampled from a BOS distribution of parameters (μ,π), with a number of levels equals to m.
Author(s)
Margot Selosse, Julien Jacques, Christophe Biernacki.

Examples

```r
library(ordinalClust)
data("dataqol")
set.seed(5)

m=7
nr=10000
mu=5
pi=0.5

probaBOS=rep(0,m)
for (im in 1:m) probaBOS[im]=pejSim(im,m,mu,pi)
M <- sample(1:m,nr,prob = probaBOS, replace=TRUE)
```

Description

Plots the result of a classification, clustering or co-clustering that were performed from the following functions: `bosclassif`, `bosclust`, `boscoclust`.

Methods

- `signature(object = "ResultClassifOrdinal")`
- `signature(object = "ResultClustOrdinal")`
- `signature(object = "ResultCoclustOrdinal")`

predict

Description

`predict` method with the result of the function `bosclassif`, and a new sample to predict the classes.

Methods

- `signature(object = "ResultClassifOrdinal")` Use this method with the result of the function `bosclassif`, and a new sample to predict the classes.
Description

Prints the result of a classification, clustering or co-clustering that were performed from the following functions: bosclassif, bosclust, boscoclust.

Methods

signature(object = "ResultClassifOrdinal")
signature(object = "ResultClustOrdinal")
signature(object = "ResultCoclustOrdinal"