Package 'BNPMIXcluster'

February 1, 2017

Type Package

Title Bayesian Nonparametric Model for Clustering with Mixed Scale Variables

Version 0.2.0

Description Bayesian nonparametric approach for clustering that is capable to combine different types of variables (continuous, ordinal and nominal) and also accommodates for different sampling probabilities in a complex survey design. The model is based on a location mixture model with a Poisson-Dirichlet process prior on the location parameters of the associated latent variables. The package performs the clustering model described in Carmona, C., Nieto-Barajas, L. E., Canale, A. (2016) <http://arxiv.org/abs/1612.00083>.

License GPL

LazyData TRUE

Depends R (>= 2.10)

Imports matrixcalc, truncnorm, mvtnorm, plyr, MASS, compiler

RoxygenNote 5.0.1

Suggests scatterplot3d

NeedsCompilation no

Author Christian Carmona [aut, cre],
Luis Nieto-Barajas [aut],
Antonio Canale [ctb]

Maintainer Christian Carmona <carmona@stats.ox.ac.uk>

Repository CRAN

Date/Publication 2017-02-01 16:12:00

R topics documented:

MIXclustering 2
plot.MIXcluster 8
poverty.data 9
sim.cluster.data 10
summary.MIXcluster 12

Index **14**

`MIXclustering` *Bayesian Nonparametric Model for Clustering with Mixed Scale Variables*

Description

`MIXclustering` is used to perform cluster analisis of individuals using a Bayesian nonparametric mixture model that jointly models mixed scale data and accommodates for different sampling probabilities. The model is described in Carmona, C., Nieto-Barajas, L. E., Canale, A. (2016).

Usage

```
MIXclustering(x, var_type, n.iter_out = 1000, n.burn = 100, n.thin = 3,
  a_fix = NULL, alpha = 0.5, d_0_a = 1, d_1_a = 1, b_fix = NULL,
  d_0_b = 1, d_1_b = 5, eta = 2, d_0_z = 2.1, d_1_z = 30, kappa = 5,
  delta = 4, d_0_mu = 2.1, d_1_mu = 30, sampling_prob = NULL,
  expansion_f = NULL, max.time = Inf)
```

Arguments

`x`	Matrix or data frame containing the data to be clustered.
`var_type`	Character vector that indicates the type of variable in each column of `x`. Three possible types: • "**c**" for continuous variables. It is assumed to be Gaussian-shaped. • "**o**" for ordinal variables (binary and ordered categorical). • "**m**" for nominal variables (non-ordered categorical).
`n.iter_out`	Number of effective iterations in the MCMC procedure for clustering.
`n.burn`	Number of iterations discarded as part of the burn-in period at the beginning MCMC procedure.
`n.thin`	Number of iterations discarded between two effective iterations, with the purpose of reducing the autocorrelation in the chain.
`a_fix`	A numeric value to set the parameter a in the model. If `NULL` (default), the parameter a is assigned a prior distribution. See `details`.
`alpha`	Hyperparameter in the prior distribution of a. See `details`.
`d_0_a`	Hyperparameter in the prior distribution of a. See `details`.
`d_1_a`	Hyperparameter in the prior distribution of a. See `details`.
`b_fix`	A numeric value to set the parameter b in the model. If `NULL` (default), the parameter b is assigned a prior distribution. See `details`.
`d_0_b`	Hyperparameter in the prior distribution of b. See `details`.
`d_1_b`	Hyperparameter in the prior distribution of b. See `details`.
`eta`	Tuning parameter controlling the proposal in the *Metropolis-Hastings* step for b.
`d_0_z`	Hyperparameter in the prior distribution of the variance for the latent variables. See `details`.

d_1_z	Hyperparameter in the prior distribution of the variance for the latent variables. See details.
kappa	Tuning parameter controlling the proposal in the *Metropolis-Hastings* step for the variance of latent variables.
delta	Tuning parameter controlling the proposal in the *Metropolis-Hastings* step for the correlation of latent variables.
d_0_mu	Hyperparameter in the prior distribution of the variance within each cluster. See details.
d_1_mu	Hyperparameter in the prior distribution of the variance within each cluster. See details.
sampling_prob	vector with the sampling probabilities π_i for each individual in case that the data come from a complex survey sample. By default $\pi_i = 1$.
expansion_f	vector with the expansion factors, the reciprocal of the sampling probabilities, $w_i = 1/\pi_i$. If both sampling_prob and expansion_f are specified, preference is given to sampling_prob.
max.time	Maximum time tolerated to be spend in the sampling procedure of the Metropolis-Hastings steps. If reached the routine is stopped with error.

Details

The model consists in a bayesian non-parametric approach for clustering that is capable to combine different types of variables through the usage of associated continuous latent variables. The clustering mechanism is based on a location mixture model with a Poisson-Dirichlet (PD) process prior on the location parameters $\mu_i; i = 1, ..., n$ of the associated latent variables.

Computational inference about the cluster allocation and the posterior distribution of the parameters are performed using MCMC simulations.

A full description of the model is in the article Carmona et al. (2016) (preprint: http://arxiv.org/abs/1612.00083). See Reference.

The model consider an individual y_i that is characterized by a multivariate response of dimension p, i.e., $y_i = (y_{i,1}, ..., y_{i,p})$. The total number of variables p is divided into c continuous variables, o ordinal variables, and m nominal variables such that $p = c + o + m$.

For each response $y_i = (y_{i,1}, ..., y_{i,p})$ (of dimension p) a corresponding latent vector $z_i = (z_{i,1}, ..., z_{i,q})$ (of dimension q) is created, according to the following:

- For each continuous variable $y_{i,j}; j = 1, ..., c$ the algorithm uses a latent with the same values $z_{i,j} = y_{i,j}$.
- For each ordinal variable $y_{i,j}, j = c + 1, ..., c + o$, with K_j different ordered values, the algorithm creates one latent $z_{i,j}$, that allows to map the categories into continuous values divided by thresholds. For example, for a binary y_j, we have $y_j = 0$ iff $z_j < 0$ and $y_j = 1$ iff $z_j > 0$
- For each nominal variable $y_{i,j}, j = c+o+1, ..., c+o+m$, with L_j categories, the algorithm require $L_j - 1$ latent variables, whose relative order is consistent with the observed category.

The data may come from a complex survey sample where each individual y_i has known sampling probability $\pi_i, i = 1, ..., n$. The reciprocal of these sampling probabilities, $w_i = 1/\pi_i$, are called expansion factors or sampling design weights.

The joint model for the latent vector is therefore:

$(z_i|\mu_i, \Sigma) \sim N_q(\mu_i, \pi_i\Sigma)$

(Note: the final model in Carmona et al. (2016) has variance $\kappa\pi_i\Sigma$. This value of κ can be used in the package through a transformed sampling probability vector $\pi_i^* = \kappa\pi_i$)

The clustering model will be based in an appropriate choice of the prior distribution on the μ_i's. A clustering of the μ_i's will induce a clustering of the y_i's. Our prior on the μ_i's will be:

$\mu_i|G \sim G$, iid for $i = 1, ..., n$

Where $G \sim PD(a, b, G_0)$ is a Poisson-Dirichlet process with parameters $a \in [0, 1)$, $b > -a$ and centring measure G_0. The Dirichlet and the normalized stable processes arise when $a = 0$ and when $b = 0$, respectively.

In consequence, this choice of prior implies that the μ_i's are exchangeable with marginal distribution $\mu_i \sim G_0$ for all $i = 1, ..., n$.

In our case, $G(\mu) = N(0, \Sigma_\mu)$, where $\Sigma_\mu = diag(\sigma^2_{\mu 1}, ..., \sigma^2_{\mu q})$.

The parameters a and b in the model define the PD process and therefore control the number of groups. These parameters can be fixed, resulting in a larger/smaller number of groups if assigned a larger/smaller value, respectively.

There are 9 hyperparameters in the function that also characterize the prior distributions in the model:

- f(a) = alpha * I(a=0) + (1-alpha) * dbeta(a | d_0_a , d_0_a)
- f(b | a) = dgamma(b + a | d_0_b , d_1_b)
- sigma^2 ~ inverse-gamma(d_0_z , d_1_z)
- sigma^2_mu ~ inverse-gamma(d_0_mu , d_1_mu)

The definition of these values also affect the number of resulting clusters since they affect the variance implied in the model.

For example, increasing the values of `d_1_a` and `d_1_b` reduce the number of groups.

Finally, the function parameters `eta`, `kappa`, `delta` are tuning parameters that control the acceptance rate in the random-walk MH steps of the new proposed values for the parameters b, $\Lambda_{j,j}$ (variance of latents) and $\Omega_{i,j}$ (correlation of latents). These parameters are not recomended to be changed (used in the internal functions: `sampling_b` , `sampling_Lambda_jj` , `sampling_Omega_ij`).

Value

`MIXclustering` returns a S3 object of class "`MIXcluster`".

The generic methods `summary` and `plot` are defined for this class.

An object of class "MIXcluster" is a list containing the following components:

`cluster` vector with the cluster allocation for each each row in the data. It corresponds to the iteration which is Closest-To-Average (CTA) arrangement.

`Y.cluster.summary` a summary of the data divided by the allocation in `$cluster`.

`Y.var_type` vector with the variable types in the data.

`Y.na` vector specifying the rows with missing values.

`Y.n` number of rows in the data.

`Y.p` number of variables in the data.

`MC.clusters` matrix with the cluster allocation for each each row in the data. Each column corresponds to an effective iteration in the MCMC simulation of the model (after discarding burn-in and thinning iterations).

`cluster.matrix.avg` average similarity matrix of size n by n.

`MC.values` a list with the simulated values of the chains for the parameters a,b,Λ,Ω.

`MC.accept.rate` a named vector with the acceptance rates for each parameter. It includes iterations that are discarded in the burn-in period and thinning.

`call` the matched call.

References

Carmona, C., Nieto-Barajas, L. E. & Canale, A. (2016). *Model based approach for household clustering with mixed scale variables.* (preprint: http://arxiv.org/abs/1612.00083)

See Also

`summary.MIXcluster` for a summary of the clustering results, `plot.MIXcluster` for graphical representation of results.

Examples

```
##### Testing "MIXclustering" function with simulated clustered data #####
# for more information about the data...
## Not run:
help(sim.cluster.data)

## End(Not run)


### Exercise (Ic) from section 5.1 in Carmona et al. (2016)
### using 3 continuous variables
## Not run:
set.seed(0) # for reproducibility

Y_data <- sim.cluster.data[,2:4]
colnames(Y_data) <- paste("Y",1:3,sep=".")

cluster_estim_Ic <- MIXclustering( Y_data,
                           var_type=c("c","c","c"),
                           n.iter_out=1000, #
                           n.burn=200,
                           n.thin=2,
                           alpha=0.5,
                           d_0_a = 1, d_1_a = 1,
                           d_0_b = 1, d_1_b = 1,
                           d_0_z = 2.1, d_1_z = 30,
                           d_0_mu = 2.1, d_1_mu = 30
```

```
                          )
cluster_estim_Ic

# Summary of clustering results
summary(cluster_estim_Ic)

# Grafical representation of clustering results
plot(cluster_estim_Ic,type="heatmap")
plot(cluster_estim_Ic,type="chain")

# Comparison with the original clusters in the simulated data
plot(x=jitter(sim.cluster.data$cluster),
     y=jitter(cluster_estim_Ic$cluster),
     main="",
     xlab="Real cluster",
     ylab="Model cluster",
     pch=19, col="#FF000020")

## End(Not run)


### Exercise (IIb) from section 5.1 in Carmona et al. (2016)
### Two binary and one continuous variables
## Not run:
set.seed(0) # for reproducibility

Y_data <- matrix(NA,nrow=nrow(sim.cluster.data),ncol=3)
colnames(Y_data) <- paste("Y",1:3,sep=".")
Y_data[,1] <- findInterval( sim.cluster.data[,2], c(-Inf,5,Inf) )-1
Y_data[,2] <- sim.cluster.data[,3]
Y_data[,3] <- findInterval( sim.cluster.data[,4], c(-Inf,3,Inf) )-1

cluster_estim_IIb <- MIXclustering( Y_data,
                          var_type=c("o","c","o"),
                          n.iter_out=1000,
                          n.burn=200,
                          n.thin=2,
                          alpha=0.5,
                          d_0_a = 1, d_1_a = 1,
                          d_0_b = 1, d_1_b = 1,
                          d_0_z = 1, d_1_z = 1,
                          d_0_mu = 1, d_1_mu = 1
                          )

summary(cluster_estim_IIb)
plot(cluster_estim_IIb,type="heatmap")
plot(cluster_estim_IIb,type="chain")

## End(Not run)


### Exercise (IIIa) from section 5.1 in Carmona et al. (2016)
### Two binary, one ordinal, and one (non-informative) continuous variables
```

```
## Not run:
set.seed(0) # for reproducibility

Y_data <- matrix(NA,nrow=nrow(sim.cluster.data),ncol=4)
colnames(Y_data) <- paste("Y",1:4,sep=".")
Y_data[,1] <- findInterval( sim.cluster.data[,2], c(-Inf,5,Inf) )-1
Y_data[,2] <- findInterval( sim.cluster.data[,3], c(-Inf,4,5,Inf) )-1
Y_data[,3] <- findInterval( sim.cluster.data[,4], c(-Inf,3,Inf) )-1
Y_data[,4] <- rnorm(n=nrow(sim.cluster.data),mean=0,sd=1)

cluster_estim_IIIa <- MIXclustering( Y_data,
                          var_type=c("o","o","o","c"),
                          n.iter_out=1000,
                          n.burn=200,
                          n.thin=2,
                          alpha=0.5,
                          d_0_a = 1, d_1_a = 1,
                          d_0_b = 1, d_1_b = 1,
                          d_0_z = 0.1, d_1_z = 0.1,
                          d_0_mu = 0.1, d_1_mu = 0.1
                          )

summary(cluster_estim_IIIa)
plot(cluster_estim_IIIa,type="heatmap")
plot(cluster_estim_IIIa,type="chain")

## End(Not run)


##### Testing "MIXclustering" function with poverty.data #####
# Using entity 15 (Edomex) #

## Not run:
Y_names <- c("ict_norm",
             "ic_ali","ic_asalud","ic_cv",
             "ic_rezedu","ic_sbv","ic_segsoc",
             "niv_ed","tam_loc")
aux_subset <- rep(T,nrow(poverty.data))
aux_subset <- aux_subset & is.element(substr(poverty.data$folioviv,1,2),"15")

Y_data <- poverty.data[aux_subset,Y_names]

sampling_prob_pov_iii <- 4 * mean(poverty.data[aux_subset,"factor_hog"])
sampling_prob_pov_iii <- sampling_prob_pov_iii * 1/poverty.data[aux_subset,"factor_hog"]

cluster_estim_pov_iii <- MIXclustering(x=Y_data,
                               var_type=c("c","o","o","o","o","o","o","m","m"),
                               n.iter_out=1000,
                               n.burn=200,
                               n.thin=2,
                               d_0_a = 1, d_1_a = 1,
                               d_0_b = 1, d_1_b = 1,
                               d_0_z = 2.1, d_1_z = 30,
```

```
                              d_0_mu = 2.1, d_1_mu = 30,
                              sampling_prob = sampling_prob_pov_iii
                              )

summary(cluster_estim_pov_iii)
plot(cluster_estim_pov_iii,type="heatmap")
plot(cluster_estim_pov_iii,type="chain")

## End(Not run)
```

`plot.MIXcluster` *Plotting clustering results for "*`MIXcluster`*" objects*

Description

Plotting method for objects inheriting from class "`MIXcluster`".

Usage

```
## S3 method for class 'MIXcluster'
plot(x, type = c("heatmap", "chain")[1],
  chain.obj = c("n.cluster", "a", "b", "Lambda", "Omega", "all")[1], ...)
```

Arguments

`x`	an object of class "`MIXcluster`"
`type`	what type of plot should be drawn. Possible types are:
	`"heatmap"` (default) draws a heatmap of the average similarity matrix for the effective iterations of the MCMC.
	`"chain"` for the evolution and histograms of the chains for parameters in the model.
`chain.obj`	if `type="chain"`, this specifies what chain will be plotted. Possible types are:
	`"n.cluster"` (default) for the number of clusters.
	`"a"` for the a parameter of the model.
	`"b"` for the b parameter of the model.
	`"Lambda"` one plot for each element in the diagonal of the Λ matrix of the model (variance of latent variables).
	`"Omega"` one plot for each element above the diagonal of the Ω matrix of the model (correlation between latent variables).
	`"all"` for all of the above.
`...`	further arguments passed to or from other methods.

See Also

`MIXclustering`

poverty.data *Poverty data for testing the* BNPMIXcluster *package*

Description

Poverty indicators observed in Mexico for 2014.

The original data is available in the file "R_2014.zip" from CONEVAL's website: http://www.coneval.org.mx/Medicion/MP/Paginas/Programas_BD_10_12_14.aspx

(download zip file directly from: http://www.coneval.org.mx/Medicion/MP/Documents/Programas_calculo_pobreza_10_12_14/R_2014.zip)

This data frame presents indicators aggregated by household. The aggregation was done by the authors acording with code in section Examples.

Usage

```
poverty.data
```

Details

poverty.data is a data frame with 58121 rows and 13 variables, with the following columns:

proyecto Data source identifier (1=MCS, 2=ENIGH)

folioviv Household identifier level 1

foliohog Household identifier level 2

ict_norm (continuous) Total income in the household (in mexican pesos).

ic_ali (binary) Indicator for deprivation to feeding: 1-yes,0-no

ic_asalud (binary) Indicator for deprivation to health services: 1-yes,0-no

ic_cv (binary) Housing quality: 1-bad, 0-good

ic_rezedu (binary) Indicator for education backwardness: 1-yes,0-no

ic_sbv (binary) Indicator for deprivation to basic public services: 1-yes,0-no

ic_segsoc (binary) Indicator for deprivation to social security: 1-yes,0-no

niv_ed (categorical, ordered) Maximum educational level in the household: 0-incomplete primary: 1-incomplete secondary, 2-complete secondary or more

tam_loc (categorical, nominal) Size of locality according to the number of people living there: 1-(100000, ∞), 2-[15000, 100000), 3-[2500, 15000), 4-[0, 2500)

factor_hog Expansion factor for the household, according to the complex survey design.

See Also

MIXclustering

Examples

```
##### Generates poverty.data using the original data from CONEVAL's website #####

## Not run:
# step 1:
#     Download and unzip the file "R_2014.zip"
#     available in:
#    http://www.coneval.org.mx/Medicion/MP/Documents/Programas_calculo_pobreza_10_12_14/R_2014.zip

# step 2:
#     extract and read the csv file "pobreza_14.csv"

coneval.poverty.data <- read.csv("pobreza_14.csv", na.strings=c("NA",""))

# step 3:
#     Execute the following code...

var_id <- c("proyecto","folioviv","foliohog","numren")
for(i in match(var_id,colnames(coneval.poverty.data)) ){
     coneval.poverty.data[,i] <- formatC( x=as.numeric(coneval.poverty.data[,i]),
                                          width=max(nchar(coneval.poverty.data[,i])),
                                          format="f",flag="0",digits=0
                                          )
}

# normalizing the continuous variable for income #
b <- quantile(coneval.poverty.data$ict,probs=0.01)
coneval.poverty.data$ict_norm <- log(coneval.poverty.data$ict+b)

# Aggregating data at household level
Y_names <- c("ict_norm",
             "ic_ali","ic_asalud","ic_cv",
             "ic_rezedu","ic_sbv","ic_segsoc",
             "niv_ed","tam_loc")
agg_form <- as.formula( paste( "cbind(",paste(c(Y_names,"factor_hog"),collapse=",") ,")",
                               "~proyecto+folioviv+foliohog"
                             )
                      )

poverty.data <- aggregate(agg_form,FUN="max",data=coneval.poverty.data)


## End(Not run)
```

`sim.cluster.data` *Simulated data for testing the* BNPMIXcluster *package*

Description

Simulated values for three continuos variables under the existence of three clusters.

The data consists of a three-variate Normal distribution with different mean and covariance matrix between clusters.

This can be assumed either as continuos data to be clustered Y=(Y_1,Y_2,Y_3); or also can be used as the underlying latent data that can be transformed into observable variables Y_i=f(Z_i), which can be continuos or categorical.

Usage

```
sim.cluster.data
```

Format

A data frame with 100 rows and 4 variables.

cluster Indicates the cluster for each row

Z1,Z2,Z3 Continuos values coming from a multivariate normal distribution, given the cluster

Details

A data frame with 100 rows and 4 variables.

See Also

MIXclustering

Examples

```
### Visualizing the simulated data for clustering ###

require(scatterplot3d)

cluster_color <- c(rgb(1,0,0,alpha = 0.5),
                   rgb(0,0,1,alpha = 0.5),
                   rgb(0,0.5,0,alpha = 0.5))
cluster_color <- cluster_color[sim.cluster.data$cluster]
cluster_pch <- c(19,15,17)[sim.cluster.data$cluster]
par(mfrow=c(2,2))
par(mar=c(4,5,2,2))

scatterplot3d::scatterplot3d(x=sim.cluster.data$Z1,y = sim.cluster.data$Z2, z=sim.cluster.data$Z3,
              color=cluster_color,pch=cluster_pch,
              xlab="Z1",ylab="Z2",zlab="Z3",
              main="Simulated data in 3 clusters"
              )
par(mar=c(4,5,2,2))
plot(sim.cluster.data[,c("Z2","Z3")],col=cluster_color,pch=cluster_pch,xlab="Z2",ylab="Z3")
par(mar=c(4,5,2,2))
```

```
plot(sim.cluster.data[,c("Z1","Z3")],col=cluster_color,pch=cluster_pch,xlab="Z1",ylab="Z3")
par(mar=c(4,5,2,2))
plot(sim.cluster.data[,c("Z1","Z2")],col=cluster_color,pch=cluster_pch,xlab="Z1",ylab="Z2")


### Code to generate the simulated data from scratch ###

require(MASS)
set.seed(0)

n.sim <- 100
n.cluster <- 3
p <- 3

mu_sim.cluster.data.Z <- matrix( c(2,2,5,
                                   6,4,2,
                                   1,6,2) ,
                            nrow=n.cluster, ncol=p, byrow=TRUE)

sigma_sim.cluster.data.Z <- array(dim=c(3,3,3))
sigma_sim.cluster.data.Z[,,1] <- diag(3)
sigma_sim.cluster.data.Z[,,2] <- matrix( c(0.1,0,0,
                                           0,2,0,
                                           0,0,0.1) ,
                            nrow=n.cluster, ncol=p, byrow=TRUE)

sigma_sim.cluster.data.Z[,,3] <- matrix( c(2,0,0,
                                           0,0.1,0,
                                           0,0,0.1) ,
                            nrow=n.cluster, ncol=p, byrow=TRUE)

sim.cluster.data <- data.frame(cluster=sample(x=1:n.cluster,size=n.sim,replace=TRUE))

sim.cluster.data.Z <- matrix(NA,nrow=n.sim,ncol=p)

for(i in unique(sim.cluster.data$cluster)) {
     sim.cluster.data.Z[sim.cluster.data[,1]==i,] <- MASS::mvrnorm(
                                                     n=sum(sim.cluster.data[,1]==i),
                                                     mu=mu_sim.cluster.data.Z[i,],
                                                     Sigma=sigma_sim.cluster.data.Z[,,i]
                                                     )
}
colnames(sim.cluster.data.Z) <- paste("Z",1:ncol(sim.cluster.data.Z),sep="")
sim.cluster.data <- cbind(sim.cluster.data,sim.cluster.data.Z)
```

---

summary.MIXcluster *Summarizing clustering results*

Description

summary method for class "MIXcluster".

Usage

```
## S3 method for class 'MIXcluster'
summary(object, ...)
```

Arguments

object	an object of class "MIXcluster"
...	further arguments passed to or from other methods.

See Also

MIXclustering

Index

MIXclustering, 2, *8*, *9*, *11*, *13*

plot, *4*
plot.MIXcluster, *5*, 8
poverty.data, 9

sim.cluster.data, 10
summary, *4*
summary.MIXcluster, *5*, 12