Package ‘HDclustVS’

November 17, 2020

Title Block-wise Variable Selection for Clustering via Latent States of Mixture Models

Version 1.0

Author Beomseok Seo [aut, cre],
Lin Lin [aut],
Jia Li [aut]

Maintainer Beomseok Seo <bzs32@psu.edu>

Description HDclustVS is an R package for new block-wise variable selection methods for clustering, HMM-VB-VS and GMM-VB-VS, which exploit the latent states of the hidden Markov model on variable blocks or Gaussian mixture model. The variable blocks are formed by early-stop-and-sorted-depth-first-search (ESS-DFS) on a dendrogram created based on the mutual information between any pair of variables. Then, the variable selection is conducted by an independence test between the latent states and semi-clusters which are the smaller clusters that will be further grouped into final clusters. This package will be merged with HDclust in CRAN in the near future.

Depends R (>= 3.1.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr,
rmarkdown,
mclust

VignetteBuilder knitr

Imports HDclust,
clusterGeneration,
dplyr,
entropy,
ggplot2,
mclust,
mvtnorm

R topics documented:

`constVB` *Variable Blocks Construction by ESS-DFS.*

Description

This function constructs variable blocks by ESS-DFS algorithm.

Usage

```
constVB(X, pwmi, max.size = dim(X)[2]/10)
```

Arguments

`X`	A data matrix.
`pwmi`	A matrix of pairwise mutual information.
`max.size`	Maximum size of variable blocks.

Value

A list of the variable indices grouped by the constructed variable blocks.

`vb`	variable indices for variable blocks.
`ix`	variable block indices for variables.

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5
# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
# Variable block construction by ESS-DFS.
vbs = constVB(X,pwmi,max.vb.size)
```

`distDB` *# Davis-Bouldin Type Distance*

Description

This function calculates Davis-Bouldin type distance.

Usage

```
distDB(mean, Sigma)
```

Arguments

`mean`	A matrix of mean values.
`Sigma`	A list of variance-covariance matrix.

Value

A distance matrix.

`distDB2` *# Davis-Bouldin Type Distance*

Description

This function calculates Davis-Bouldin type distance from the estimated GMM components of the output of HMM-VB.

Usage

```
distDB2(X, hmmvb, clust)
```

Arguments

`X`	A data matrix.
`hmmvb`	An object of class 'HMMVB'
`clust`	An object of class 'HMMVBclust'

Value

A distance matrix.

`finalcls` *Clustering Latent State Configurations by BFS Search.*

Description

This function finds the cluster indices from HMM-VB or GMM-VB output.

Usage

```
finalcls(X, fit, C, min.size.cls = 5)
```

Arguments

`X`	A data matrix.
`fit`	HMM-VB or GMM-VB output.
`C`	The number of clusters.

Value

A vector of cluster indices.

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5

# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
# Variable block construction by ESS-DFS.
vbs = constVB(X,pwmi,max.vb.size)

# Fitting HMM-VB with 5 components.
fit = fitHmmvb(X,C,vbs)

# Semi-clusters
```

```
semi.cls = semicls(X,fit)
# Variable block selection by a bimodality test.
chosen.vb = semi.cls$chosen.vb
# Reduce the model structure
red.dat = reduceVB(X,fit,chosen.vb)
red.X = red.dat$X
red.vbs = red.dat$vbs

# Re-estimation of the HMM-VB model with reduced dimensions
re.fit = fitHmmvb(red.X,C,red.vbs)
# Final clustering
final.cls = finalcls(red.dat$X,re.fit,C,5)
```

`fitGmmvb` *Gaussian Mixture Model on Variable Blocks (GMM-VB).*

Description

This function fits the model GMM-VB on a given data set.

Usage

```
fitGmmvb(X, C, vbs, numst = rep(0, length(vbs$vb)))
```

Arguments

X	A data matrix.
C	The number of clusters.
vbs	Variable block structure. The output of constVB.

Value

A list of estimated parameters for GMM-VB.

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5

# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
# Variable block construction by ESS-DFS.
vbs = constVB(X,pwmi,max.vb.size)

# Fitting HMM-VB with 5 components.
```

```
fit = fitGmmvb(X,C,vbs)

# Semi-clusters
semi.cls = semicls(X,fit)
# Variable block selection by a bimodality test.
chosen.vb = semi.cls$chosen.vb
# Reduce the model structure
red.dat = reduceVB(X,fit,chosen.vb)
red.X = red.dat$X
red.vbs = red.dat$vbs

# Re-estimation of the HMM-VB model with reduced dimensions
re.fit = fitGmmvb(red.X,C,red.vbs)
# Final clustering
final.cls = finalcls(red.dat$X,re.fit,C,5)
#' @export
```

`fitHmmvb` *Hidden Markov Model on Variable Blocks (HMM-VB).*

Description

This function fitt the model HMM-VB on a given data set.

Usage

```
fitHmmvb(X, C, vbs, numst = rep(0, length(vbs$vb)))
```

Arguments

X	A data matrix.
C	The number of clusters.
vbs	Variable block structure. The output of constVB.

Value

A list of estimated parameters for HMM-VB.

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5

# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
```

```
# Variable block construction by ESS-DFS.
vbs = constVB(X,pwmi,max.vb.size)

# Fitting HMM-VB with 5 components.
fit = fitHmmvb(X,C,vbs)

# Semi-clusters
semi.cls = semicls(X,fit)
# Variable block selection by a bimodality test.
chosen.vb = semi.cls$chosen.vb
# Reduce the model structure
red.dat = reduceVB(X,fit,chosen.vb)
red.X = red.dat$X
red.vbs = red.dat$vbs

# Re-estimation of the HMM-VB model with reduced dimensions
re.fit = fitHmmvb(red.X,C,red.vbs)
# Final clustering
final.cls = finalcls(red.dat$X,re.fit,C,5)
```

---

genData *Generating multi-dimensional Gaussian random partition.*

Description

This function generates multi-dimensional Gaussian random partition with noisy variables for simulation study.

Usage

```
genData(
  n,
  p1,
  p2,
  C,
  relev.var = c(10, 50),
  irrel.var = c(100, 400),
  overlap = FALSE,
  BClevel = 10^-5,
  MinSize = 1/(5 * C),
  rep = 1
)
```

Arguments

`n`	Size of observations.
`p1`	Dimension of relevant variables.
`p2`	Dimension of irrelevant variables.
`C`	The number of clusters.
`relev.var`	Range of positive number. Default is [10,50]. Range for variances of relevant variables.

`irrel.var`	Range of positive number. Default is [100,400]. Range for variances of irrelevant variables.
`overlap`	Logical. If it is True, the algorithm allows clusters on relevant variable space to overlap each other. Overlap is evaluated by Bhattacharyya coefficient.
`BClevel`	Threshold of Bhattacharyya coefficient. Default is 10^-5.
`MinSize`	Minimum proportion allowed for each cluster. Default is 1/(5*C).

Value

A synthesized data matrix.

Examples

```
sim1 = genData(n=300,p1=100,p2=100,C=5,rep=1)
sim2 = genData2(n=300,p1=100,p2=100,C=5,rep=1)
```

genData2 *Generating multi-dimensional Gaussian partition based on given parameters.*

Description

This function generates multi-dimensional Gaussian partition with noisy variables for simulation study based on pre-specified mean and covariance matrix parameters.

Usage

```
genData2(n, p1, p2, C, rep = 10)
```

Arguments

`n`	Size of observations.
`p1`	Dimension of relevant variables.
`p2`	Dimension of irrelevant variables.
`C`	The number of clusters.
`rep`	The number of data sets.

Value

A list of the synthesized data matrix.

Examples

```
sim1 = genData(n=300,p1=100,p2=100,C=5,rep=1)
sim2 = genData2(n=300,p1=100,p2=100,C=5,rep=1)
```

genData3 *Generating multi-dimensional Gaussian random partition.*

Description

This function generates multi-dimensional Gaussian random partition with noisy variables for simulation study.

Usage

```
genData3(
  n,
  relev.blocks,
  irrel.blocks,
  C,
  relev.var = c(100, 200),
  irrel.var = c(100, 200),
  df = 4,
  overlap = FALSE,
  BClevel = 10^-5,
  MinSize = 1/(5 * C),
  rep = 1
)
```

Arguments

`n`	Size of observations.
`relev.blocks`	A vector of positive integers indicating the size of each relevant block.
`irrel.blocks`	A vector of positive integers indicating the size of each irrelevant block.
`C`	The number of clusters.
`relev.var`	Range of positive number. Default is [10,50]. Range for variances of relevant variables.
`irrel.var`	Range of positive number. Default is [100,400]. Range for variances of irrelevant variables.
`overlap`	Logical. If it is True, the algorithm allows clusters on relevant variable space to overlap each other. Overlap is evaluated by Bhattacharyya coefficient.
`BClevel`	Threshold of Bhattacharyya coefficient. Default is 10^-5.
`MinSize`	Minimum proportion allowed for each cluster. Default is 1/(5*C).

Value

A synthesized data matrix.

Examples

```
sim1 = genData(n=300,p1=100,p2=100,C=5,rep=1)
sim2 = genData2(n=300,p1=100,p2=100,C=5,rep=1)
sim3 = genData3(n=1000,relev.blocks=c(200,200,200,200,200),irrel.blocks=1000,relev.var=c(50,50),irrel.var=c
```

`LPS` *A single-cell RNA-sequencing data from primary mamalian cells*

Description

A dataset consisting of 131 primary mammalian cells with 534 genes. This data set contains only two cell clusters: unstimulated and stimulated cells.

Usage

```
LPS
```

Format

A list containing four components \itemcellcell names \itemcell.stagetrue labels of cells \itemg-namesgene names \itemexprexpression data

Source

https://www.ncbi.nlm.nih.gov/pubmed/19729616

`MEF` *A single-cell RNA-sequencing data from mouse cells.*

Description

A dataset consisting of 92 single cells with 35 genes from two different mouse cell types: embryonic stem cells (ES R1) and embryonic fibroblasts (MEFs).

Usage

```
MEF
```

Format

A list containing four components \itemcellcell names \itemcell.stagetrue labels of cells \itemg-namesgene names \itemexprexpression data

Source

https://www.ncbi.nlm.nih.gov/pubmed/19729616

nodeSize *Computing the size of the subtree of a node.*

Description

This function calculates the size of leaves-set of a node.

Usage

```
nodeSize(M)
```

Arguments

`M`	A matrix of merges of nodes, which is obtained from hclust() output.

Value

A vector with the size of leaves-set of each node.

pairwiseMI *Pairwise Mutual Information.*

Description

This function calculates pairwise mutual information between variables.

Usage

```
pairwiseMI(X, numBins1 = 10, numBins2 = 10)
```

Arguments

`X`	A data matrix.

Value

A matrix of pairwise mutual information.

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5
# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
```

plotCls *Visulization of a partition on 2 dimensional space with convex hull.*

Description

This function plots a partition on 2 dimensional principal components space with convex hull.

Usage

```
plotCls(
  X,
  z,
  title = "",
  xlab = "",
  ylab = "",
  legend.title = "",
  no.legend = F
)
```

Arguments

X	A matrix of data.
z	Cluster labels of data.

Value

None

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]

PCA = prcomp(dat[[1]]$X_relev)
Xr = prcomp(dat[[1]]$X_relev)$x[,1:2]
plotCls(Xr,Y,title="Relevant var.")
Xi = prcomp(dat[[1]]$X_irrel)$x[,1:2]
plotCls(Xi,Y,title="Irrelevant var.")
Xt = prcomp(dat[[1]]$X_total)$x[,1:2]
plotCls(Xt,Y,title="Total var.")
```

reduceVB *Dimension-reduced Data Set.*

Description

This fuction computes the dimension-reduced data set.

Usage

```
reduceVB(X, fit, chosen.vb)
```

Arguments

`X`	A data matrix.
`fit`	HMM-VB or GMM-VB output.
`chosen.vb`	A vector of chosen variable block indices.

Value

A list of dimension-reduced data set and its variable block structure.

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5

# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
# Variable block construction by ESS-DFS.
vbs = constVB(X,pwmi,max.vb.size)

# Fitting HMM-VB with 5 components.
fit = fitHmmvb(X,C,vbs)

# Semi-clusters
semi.cls = semicls(X,fit)
# Variable block selection by a bimodality test.
chosen.vb = semi.cls$chosen.vb
# Reduce the model structure
red.dat = reduceVB(X,fit,chosen.vb)
red.X = red.dat$X
red.vbs = red.dat$vbs

# Re-estimation of the HMM-VB model with reduced dimensions
re.fit = fitHmmvb(red.X,C,red.vbs)
```

```
# Final clustering
final.cls = finalcls(red.dat$X,re.fit,C,5)
```

rtmvt *Generating multi-dimensional Gaussian random partition.*

Description

This function generates multi-dimensional Gaussian random partition with noisy variables for simulation study.

Usage

```
rtmvt(
  n,
  mean,
  sigma,
  df = 3,
  type = "shifted",
  lower = rep(-inf, length(mean)),
  upper = rep(inf, length(mean))
)
```

Arguments

`n`	Size of observations.
`mean`	Mean vectors.
`sigma`	Covariance matrix.
`df`	Degree of freedom.
`type`	Type of the noncentral multivariate t distribution. Refer to mvtnorm::rmvt().
`lower`	Lower bound of truncated t-distribution.
`upper`	Upper bound of truncated t-distribution.

Value

truncated t-distribution samples

search.cls *Clustering Latent State Configurations by BFS Search.*

Description

This function searches clusters by BFS search from a dendrogram tree.

Usage

```
search.cls(tree, dist, num.cls = NULL, min.size = 5)
```

Arguments

`tree`	An object of hclust result.
`dist`	The distance measures between nodes for the tree.
`num.cls`	The number of cluster. If it is NULL, it finds semi-cluster. If it is specified as an integer, it finds clusters with the number of groups.
`min.size`	The minimum size of each cluster. Default is 5.

Value

A vector of cluster indices for each leaf node. DD = dist(X) min.size.cls = 5 tree = hclust(DD) cls.ix = search.cls(tree,DD,C,min.size.cls)

`searchTreeVB` *Search Tree Graph for Variable Block Construction.*

Description

Functions used to construct variable blocks.

Usage

```
search.parent(M, loc)

search.leaves(M, loc)

search.rootnode(M, loc, max.size, dist.measure, subgraph = F)

search.vb(M, seq)
```

Arguments

`M`	A matrix of merges which is obtained from hclust() output.
`loc`	An index or a sequence of indices of nodes in the tree.
`max.size`	Maximum size of variable blocks.
`dist.measure`	Distance measure of the tree.
`seq`	A sequence of root nodes of subtrees corresponding to variable blocks

Value

search.parent gives a vector of parent node indices of given node indices in a tree.

search.leaves gives a vector of leaves sets of a given node index in a tree.

search.rootnode gives a vector of root node indices for variable blocks given the first node (loc), maximum size of block (max.size), and distance measure (dist).

search.vb gives a list of leaves sets for each root node (seq).

Examples

```
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5

# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)

DD = as.dist(t(1/pwmi))
hclmi = hclust(DD, method = 'complete')
M = hclmi$merge
node.size = nodeSize(M)
candi.node = which(node.size <= max.vb.size)
fst.rootnode = candi.node[which.max(search.parent(M,candi.node))]  # find initial root node
seq.rootnode = search.rootnode(M,fst.rootnode,max.vb.size,DD)      # root node index of each block
vb = search.vb(M,seq.rootnode)
```

`semicls` *Semi-clusters*

Description

This fuction computes semi-clusters.

Usage

```
semicls(X, fit, alpha = 0.05)
```

Arguments

`X`	A data matrix.
`fit`	HMM-VB or GMM-VB output.
`alpha`	A threshold of normalized mutual information (NMI) to classify variable blocks into irrelevant variable blocks.

Value

A list of semi-clusters and chosen variable block index.

`semi.cls.ix`	Semi-cluster indices.
`chosen.vb`	Chosen relevant variable block index set.
`normalized.mi`	Normalized mutual information between the semi-clusters and the latent states of variable blocks.
`min.size`	Minimum size of formed semi-clusters (eta).
`num.of.cls`	The number of formed semi-clusters (gamma).
`pvalue`	P-value of the bimodality test of NMI between latent states and semi-clusters

Examples

```
# Data generation
set.seed(1)
dat = genData2(n=300,p1=100,p2=100,C=5,rep=1)
X = dat[[1]]$X_total
Y = dat[[1]]$z
n = dim(X)[1];p = dim(X)[2]
# The number of clusters.
C = 5

# Maximum block size is set 5% of the total dimension.
max.vb.size = 10
# Calculate Mutual Information.
pwmi = pairwiseMI(X)
# Variable block construction by ESS-DFS.
vbs = constVB(X,pwmi,max.vb.size)

# Fitting HMM-VB with 5 components.
fit = fitHmmvb(X,C,vbs)

# Semi-clusters
semi.cls = semicls(X,fit)
# Variable block selection by a bimodality test.
chosen.vb = semi.cls$chosen.vb
# Reduce the model structure
red.dat = reduceVB(X,fit,chosen.vb)
red.X = red.dat$X
red.vbs = red.dat$vbs

# Re-estimation of the HMM-VB model with reduced dimensions
re.fit = fitHmmvb(red.X,C,red.vbs)
# Final clustering
final.cls = finalcls(red.dat$X,re.fit,C,5)
```

`sim1` *Simulated toy data*

Description

A dataset containing 300 samples and 200 variables.

Usage

```
sim1
```

Format

A list containing four components \itemX_totaltotal data \itemX_relevrelevant data \itemX_irrelirrelevant data \itemztrue labels of clusters

viterbi	*Cross Validation for Variable Selection Threshold*

Description

This function finds a threshold for variable selection using cross validation.

Usage

```
TransProb(X, fit)

EmissProb(X, fit)

Viterbi(TP, EP, NN, TT, KK)

GaussParam(HC, VV, BD, NN, TT, KK)
```

Arguments

`X`	A data matrix
`fit`	HMM-VB or GMM-VB output
`TP`	Transition probability list. The output of TransProb().
`EP`	Emission probability list. The output of EmissProb().
`NN`	Sample size. dim(X)[1].
`TT`	The length of variable blocks.
`KK`	A vector of the number of components for variable blocks.
`HC`	Hidden Markov chain
`VV`	Viterbi path. The output of Viterbi().
`BD`	A vector of the number of variables for each variable block.

Value

TransProb gives a list of transition probability.

EmissProb gives a list of emission probability.

Viterbi gives a matrix of viterbi paths.

GaussParm gives a list of estimated mixture model parameters for each sample.

`WW`	Mean values of each sample corresponding to MAP state configuration.
`SS`	Variance values of each sample corresponding to MAP state configuration.

`YAN` *A single-cell RNA-sequencing data from human preimplantation embryos and human embryonic stem cells.*

Description

A dataset consisting of 124 individual cells with 3,840 genes from human preimplantation embryos and human embryonic stem cells. There are 7 cell clusters: Oocyte and Zygote, 2-cell, 4-cell, 8-cell, Morula, Late blastocyst, hESC.

Usage

```
YAN
```

Format

A list containing four components \itemcellcell names \itemcell.stagetrue labels of cells \itemg-namesgene names \itemexprexpression data

Source

https://www.ncbi.nlm.nih.gov/pubmed/23934149

Index