Package ‘MixSim’

November 6, 2021

Version 1.1-5
Date 2021-11-06
Title Simulating Data to Study Performance of Clustering Algorithms
Depends R (>= 4.0.0), MASS
Enhances mclust, cluster
LazyLoad yes
Description The utility of this package is in simulating mixtures of Gaussian distributions with different levels of overlap between mixture components. Pairwise overlap, defined as a sum of two misclassification probabilities, measures the degree of interaction between components and can be readily employed to control the clustering complexity of datasets simulated from mixtures. These datasets can then be used for systematic performance investigation of clustering and finite mixture modeling algorithms. Among other capabilities of 'MixSim', there are computing the exact overlap for Gaussian mixtures, simulating Gaussian and non-Gaussian data, simulating outliers and noise variables, calculating various measures of agreement between two partitionings, and constructing parallel distribution plots for the graphical display of finite mixture models.
License GPL (>= 2)
NeedsCompilation yes
Maintainer Wei-Chen Chen <wccsnow@gmail.com>
Author Volodymyr Melnykov [aut], Wei-Chen Chen [aut, cre], Ranjan Maitra [aut], Robert Davies [ctb] (quadratic form probabilities), Stephen Moshier [ctb] (eigenvalue calculations), Rouben Rostamian [ctb] (memory allocation)
Repository CRAN
Date/Publication 2021-11-06 17:00:02 UTC
**MixSim-package**

**R topics documented:**

- MixSim-package .................................................. 2
- ClassProp .......................................................... 3
- MixGOM ............................................................. 4
- MixSim ............................................................... 6
- overlap ............................................................. 8
- overlapGOM ......................................................... 9
- pdplot .............................................................. 10
- perms ............................................................... 12
- print.object ......................................................... 12
- RandIndex ........................................................ 13
- simdataset .......................................................... 15
- VarInf ............................................................... 17

**Index**

<table>
<thead>
<tr>
<th>MixSim-package</th>
<th>Simulation of Gaussian Finite Mixture Models</th>
</tr>
</thead>
</table>

**Description**

Simulation of Gaussian finite mixture models for prespecified levels of average or/and maximum overlap. Pairwise overlap is defined as the sum of two misclassification probabilities.

**Details**

- Package: MixSim
- Type: Package
- Date: 2012-08-12
- License: GPL (>= 2)

Function 'MixSim' simulates a finite mixture model for a prespecified level of average or/and maximum overlap.

Function 'overlap' computes all misclassification probabilities for a finite mixture model.

Function 'pdplot' constructs a parallel distribution plot for a finite mixture model.

Function 'simdataset' simulates a dataset from a finite mixture model.

**Author(s)**

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

Maintainer: Volodymyr Melnykov <vmelnykov@cba.ua.edu>
References


Examples

```r
# Simulate parameters of a mixture model
A <- MixSim(BarOmega = 0.01, MaxOmega = 0.10, K = 10, p = 5)

# Display the mixture via the parallel distribution plot
pdplot(A$Pi, A$Mu, A$S, MaxInt = 0.5)
```

ClassProp

<table>
<thead>
<tr>
<th>ClassProp</th>
<th>Classification Proportion</th>
</tr>
</thead>
</table>

Description

Computes the agreement proportion between two classification vectors.

Usage

```r
ClassProp(id1, id2)
```

Arguments

- `id1`: first partitioning vector.
- `id2`: second partitioning vector.

Value

Returns the value of the proportion of agreeing elements.

Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.
MixGOM

References

See Also
RandIndex, and VarInf.

Examples

```r
id1 <- c(rep(1, 50), rep(2, 100))
id2 <- rep(1:3, each = 50)
ClassProp(id1, id2)
```

MixGOM

Mixture Simulation based on generalized overlap of Maitra

Description
Generates a finite mixture model with Gaussian components for a prespecified level of goMega (generalized overlap of Maitra).

Usage

```r
MixGOM(goMega = NULL, K, p, sph = FALSE, hom = FALSE,
        ecc = 0.90, PiLow = 1.0, int = c(0.0, 1.0), resN = 100,
        eps = 1e-06, lim = 1e06)
```

Arguments

goMega value of desired generalized overlap of Maitra.
K number of components.
p number of dimensions.
sph covariance matrix structure (FALSE - non-spherical, TRUE - spherical).
hom heterogeneous or homogeneous clusters (FALSE - heterogeneous, TRUE - homogeneous).
ecc maximum eccentricity.
PiLow value of the smallest mixing proportion (if 'PiLow' is not reachable with respect to K, equal proportions are taken; PiLow = 1.0 implies equal proportions by default).
int mean vectors are simulated uniformly on a hypercube with sides specified by int = (lower.bound, upper.bound).
resN maximum number of mixture resimulations.
eps error bound for overlap computation.
lim maximum number of integration terms (Davies, 1980).
MixGOM

Details

Returns mixture parameters satisfying the prespecified level of goMega.

Value

- $\mathbf{P}$: vector of mixing proportions.
- $\mathbf{M}$: matrix consisting of components’ mean vectors ($K \times p$).
- $\mathbf{S}$: set of components’ covariance matrices ($p \times p \times K$).
- goMega: value of generalized overlap of Maitra.
- fail: flag value; 0 represents successful mixture generation, 1 represents failure.

Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

References


See Also

overlapGOM, MixSim, and simdataset.

Examples

```r
set.seed(1234)

# controls average and maximum overlaps
(ex.1 <- MixGOM(goMega = 0.05, K = 4, p = 5))

# controls maximum overlap
(ex.2 <- MixGOM(goMega = 0.15, K = 4, p = 5, sph = TRUE))
```
MixSim

Mixture Simulation

Description

Generates a finite mixture model with Gaussian components for prespecified levels of maximum and/or average overlaps.

Usage

MixSim(BarOmega = NULL, MaxOmega = NULL, K, p, sph = FALSE, hom = FALSE, ecc = 0.90, PiLow = 1.0, int = c(0.0, 1.0), resN = 100, eps = 1e-06, lim = 1e06)

Arguments

- **BarOmega**: value of desired average overlap.
- **MaxOmega**: value of desired maximum overlap.
- **K**: number of components.
- **p**: number of dimensions.
- **sph**: covariance matrix structure (FALSE - non-spherical, TRUE - spherical).
- **hom**: heterogeneous or homogeneous clusters (FALSE - heterogeneous, TRUE - homogeneous).
- **ecc**: maximum eccentricity.
- **PiLow**: value of the smallest mixing proportion (if 'PiLow' is not reachable with respect to K, equal proportions are taken; PiLow = 1.0 implies equal proportions by default).
- **int**: mean vectors are simulated uniformly on a hypercube with sides specified by int = (lower.bound, upper.bound).
- **resN**: maximum number of mixture resimulations.
- **eps**: error bound for overlap computation.
- **lim**: maximum number of integration terms (Davies, 1980).

Details

If 'BarOmega' is not specified, the function generates a mixture solely based on 'MaxOmega'; if 'MaxOmega' is not specified, the function generates a mixture solely based on 'BarOmega'.

Value

- **Pi**: vector of mixing proportions.
- **Mu**: matrix consisting of components’ mean vectors (K * p).
- **S**: set of components’ covariance matrices (p * p * K).
**MixSim**

OmegaMap  
matrix of misclassification probabilities (K * K); OmegaMap[i,j] is the probability that X coming from the i-th component is classified to the j-th component.

BarOmega  
value of average overlap.

MaxOmega  
value of maximum overlap.

rcMax  
row and column numbers for the pair of components producing maximum overlap 'MaxOmega'.

fail  
flag value; 0 represents successful mixture generation, 1 represents failure.

**Author(s)**

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

**References**


**See Also**

overlap, pdplot, and simdataset.

**Examples**

```r
set.seed(1234)

# controls average and maximum overlaps
(ex.1 <- MixSim(BarOmega = 0.05, MaxOmega = 0.15, K = 4, p = 5))
summary(ex.1)

# controls average overlap
(ex.2 <- MixSim(BarOmega = 0.05, K = 4, p = 5, hom = TRUE))
summary(ex.2)

# controls maximum overlap
(ex.3 <- MixSim(MaxOmega = 0.15, K = 4, p = 5, sph = TRUE))
summary(ex.3)
```

### Description
Computes misclassification probabilities and pairwise overlaps for finite mixture models with Gaussian components. Overlap is defined as sum of two misclassification probabilities.

### Usage

```r
overlap(Pi, Mu, S, eps = 1e-06, lim = 1e06)
```

### Arguments
- `Pi` vector of mixing proportions (length K).
- `Mu` matrix consisting of components’ mean vectors (K * p).
- `S` set of components’ covariance matrices (p * p * K).
- `eps` error bound for overlap computation.
- `lim` maximum number of integration terms (Davies, 1980).

### Value
- `OmegaMap` matrix of misclassification probabilities (K * K); `OmegaMap[i,j]` is the probability that X coming from the i-th component is classified to the j-th component.
- `BarOmega` value of average overlap.
- `MaxOmega` value of maximum overlap.
- `rcMax` row and column numbers for the pair of components producing maximum overlap ‘MaxOmega’.

### Author(s)
Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

### References

### See Also
- `MixSim`, `pdplot`, and `simdataset`.
Examples

```r
data("iris", package = "datasets")
p <- ncol(iris) - 1
id <- as.integer(iris[, 5])
K <- max(id)

# estimate mixture parameters
Pi <- prop.table(tabulate(id))
Mu <- t(sapply(1:K, function(k) { colMeans(iris[id == k, -5]) }))
S <- sapply(1:K, function(k) { var(iris[id == k, -5]) })
dim(S) <- c(p, p, K)

overlap(Pi = Pi, Mu = Mu, S = S)
```

---

**overlapGOM**

*Generalized overlap of Maitra*

### Description

Computes the generalized overlap as defined by R. Maitra.

### Usage

```r
overlapGOM(Pi, Mu, S, eps = 1e-06, lim = 1e06)
```

### Arguments

- **Pi**: vector of mixing proportions (length K).
- **Mu**: matrix consisting of components’ mean vectors (K * p).
- **S**: set of components’ covariance matrices (p * p * K).
- **eps**: error bound for overlap computation.
- **lim**: maximum number of integration terms (Davies, 1980).

### Value

Returns the value of goMega.

### Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.
pdplot

Parallel Distribution Plot

Description

Constructs a parallel distribution plot for Gaussian finite mixture models.

Usage

pdplot(Pi, Mu, S, file = NULL, Nx = 5, Ny = 5, MaxInt = 1, marg = c(2,1,1,1))

Arguments

Pi vector of mixing proportions.
Mu matrix consisting of components’ mean vectors (K * p).
S set of components’ covariance matrices (p * p * K).
file name of .pdf-file.

References


See Also

MixSim, MixGOM, and overlap.

Examples

data("iris", package = "datasets")
p <- ncol(iris) - 1
id <- as.integer(iris[, 5])
K <- max(id)

# estimate mixture parameters
Pi <- prop.table(tabulate(id))
Mu <- t(sapply(1:K, function(k){ colMeans(iris[id == k, -5]) }))
S <- sapply(1:K, function(k){ var(iris[id == k, -5]) })
dim(S) <- c(p, p, K)
overlapGOM(Pi = Pi, Mu = Mu, S = S)
pdplot

Nx number of color levels for smoothing along the x-axis.
Ny number of color levels for smoothing along the y-axis.
MaxInt maximum color intensity.
marg plot margins.

Details

If 'file' is specified, produced plot will be saved as a .pdf-file.

Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

References


See Also

MixSim, overlap, and simdataset.

Examples

data(“iris”, package = “datasets”)
p <- ncol(iris) - 1
id <- as.integer(iris[, 5])
K <- max(id)

# estimate mixture parameters
Pi <- prop.table(tabulate(id))
Mu <- t(sapply(1:K, function(k){ colMeans(iris[id == k, -5]) }))
S <- sapply(1:K, function(k){ var(iris[id == k, -5]) })
dim(S) <- c(p, p, K)

pdplot(Pi = Pi, Mu = Mu, S = S)
perms

Permutations

Description
Returns all possible permutations given the number of elements.

Usage
perms(n)

Arguments
n Number of elements.

Value
Returns a matrix containing all possible permutations.

Author(s)
Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

See Also
ClassProp.

Examples
perms(3)

print.object

Functions for Printing or Summarizing Objects

Description
A MixSim and MixGOM classes are declared, and these are functions to print and summarize objects.

Usage

```r
# S3 method for class 'MixSim'
print(x, ...)
# S3 method for class 'MixSim'
summary(object, ...)
# S3 method for class 'MixGOM'
print(x, ...)
```
RandIndex

Arguments

- x: an object with the 'MixSim' (or 'MixGOM') class attributes.
- object: an object with the 'MixSim' (or 'MixGOM') class attributes.
- ...: other possible options.

Details

These are useful functions for summarizing and debugging.

For other functions, they only show summaries of objects. Use names or str to explore the details.

Value

The results will cat or print on the STDOUT by default.

Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

References


See Also

MixSim.

Examples

```r
## Not run:
# Functions applied by directly type the names of objects.

## End(Not run)
```

RandIndex

Rand’s Index

Description

Computes Rand, adjusted Rand, Fowlkes and Mallows, and Merkin indices.

Usage

RandIndex(id1, id2)
RandIndex

Arguments

  id1  first partitioning vector.
  id2  second partitioning vector.

Value

  R    Rand’s index.
  AR   adjusted Rand’s index.
  F    Fowlkes and Mallows index.
  M    Mirkin metric.

Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

References


See Also

MixSim, pdplot, simdataset, ClassProp, and VarInf.

Examples

  id1 <- c(rep(1, 50), rep(2,100))
  id2 <- rep(1:3, each = 50)
  RandIndex(id1, id2)
**simdataset**

**Dataset Simulation**

**Description**

Simulates a datasets of sample size n given parameters of finite mixture model with Gaussian components.

**Usage**

```r
simdataset(n, Pi, Mu, S, n.noise = 0, n.out = 0, alpha = 0.001, max.out = 100000, int = NULL, lambda = NULL)
```

**Arguments**

- `n` sample size.
- `Pi` vector of mixing proportions (length K).
- `Mu` matrix consisting of components’ mean vectors (K * p).
- `S` set of components’ covariance matrices (p * p * K).
- `n.noise` number of noise variables.
- `n.out` number of outlying observations.
- `alpha` level for simulating outliers.
- `max.out` maximum number of trials to simulate outliers.
- `int` interval for noise and outlier generation.
- `lambda` inverse Box-Cox transformation coefficients.

**Details**

The function simulates a dataset of n observations from a mixture model with parameters 'Pi' (mixing proportions), 'Mu' (mean vectors), and 'S' (covariance matrices). Mixture component sample sizes are produced as a realization from a multinomial distribution with probabilities given by mixing proportions. To make a dataset more challenging for clustering, a user might want to simulate noise variables or outliers. Parameter 'n.noise' specifies the desired number of noise variables. If an interval 'int' is specified, noise will be simulated from a Uniform distribution on the interval given by 'int'. Otherwise, noise will be simulated uniformly between the smallest and largest coordinates of mean vectors. 'n.out' specifies the number of observations outside (1 - 'alpha') ellipsoidal contours for the weighted component distributions. Outliers are simulated on a hypercube specified by the interval 'int'. A user can apply an inverse Box-Cox transformation providing a vector of coefficients 'lambda'. The value 1 implies that no transformation is needed for the corresponding coordinate.

**Value**

- `X` simulated dataset (n + n.out) x (p + n.noise); noise coordinates are provided in the last n.noise columns.
- `id` classification vector (length n + n.out); 0 represents an outlier.
Author(s)
Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

References


See Also
MixSim, overlap, and pdplot.

Examples
## Not run:
set.seed(1234)

repeat{
  Q <- MixSim(BarOmega = 0.01, K = 4, p = 2)
  if (Q$fail == 0) break
}

# simulate a dataset of size 300 and add 10 outliers simulated on (0,1)x(0,1)
A <- simdataset(n = 500, Pi = Q$Pi, Mu = Q$Mu, S = Q$S, n.out = 10, int = c(0, 1))
colors <- c("red", "green", "blue", "brown", "magenta")
plot(A$X, xlab = "x1", ylab = "x2", type = "n")
for (k in 0:4){
  points(A$X[A$id == k, ], col = colors[k+1], pch = 19, cex = 0.5)
}

repeat{
  Q <- MixSim(MaxOmega = 0.1, K = 4, p = 1)
  if (Q$fail == 0) break
}

# simulate a dataset of size 300 with 1 noise variable
A <- simdataset(n = 300, Pi = Q$Pi, Mu = Q$Mu, S = Q$S, n.noise = 1)
plot(A$X, xlab = "x1", ylab = "x2", type = "n")
for (k in 1:4){
  points(A$X[A$id == k, ], col = colors[k+1], pch = 19, cex = 0.5)
}

## End(Not run)
Description

Computes the variation of information for two classification vectors.

Usage

VarInf(id1, id2)

Arguments

- id1: first partitioning vector.
- id2: second partitioning vector.

Value

Returns the variation of information. It is equal to 0 if and only if two classification vectors are identical.

Author(s)

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra.

References


See Also

ClassProp, and RandIndex.

Examples

```r
id1 <- c(rep(1, 50), rep(2, 100))
id2 <- rep(1:3, each = 50)
VarInf(id1, id2)
```
Index

* cluster
  ClassProp, 3
  MixGOM, 4
  MixSim, 6
  overlap, 8
  overlapGOM, 9
  pdplot, 10
  perms, 12
  RandIndex, 13
  simdataset, 15
  VarInf, 17
* datagen
  MixGOM, 4
  MixSim, 6
  simdataset, 15
* hplot
  pdplot, 10
* programming
  print.object, 12
  Ø0_MixSim-package (MixSim-package), 2
  ClassProp, 3
  MixGOM, 4
  MixSim, 6, 13
  MixSim-package, 2
  overlap, 8
  overlapGOM, 9
  pdplot, 10
  perms, 12
  print.MixGOM (print.object), 12
  print.MixSim (print.object), 12
  print.object, 12
  RandIndex, 13
  simdataset, 15
  summary.MixSim (print.object), 12
  VarInf, 17