Package ‘MixSim’

February 19, 2015

Version 1.0-9
Date 2013-12-07
Title Simulating Data to Study Performance of Clustering Algorithms
Depends R (>= 2.14.0), MASS
LazyLoad yes
LazyData yes
Description MixSim allows 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)
Author Volodymyr Melnykov [aut, cre], Wei-Chen Chen [aut], Ranjan Maitra [aut]
Maintainer Volodymyr Melnykov <vmelnykov@cba.ua.edu>
NeedsCompilation yes
Repository CRAN
Date/Publication 2013-12-10 18:27:05
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
ClassProp


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)
```

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.

References

See Also
RandIndex, and VarInf.

Examples

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

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).
MixSim

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).
- **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)
```
Description

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

Usage

\[
\text{overlap}(\pi, \mu, S, \epsilon = 1e^{-06}, \text{lim} = 1e06)
\]

Arguments

- **\(\pi\)**: vector of mixing proportions (length \(K\)).
- **\(\mu\)**: matrix consisting of components’ mean vectors (\(K \times p\)).
- **\(S\)**: set of components’ covariance matrices (\(p \times p \times K\)).
- **\(\epsilon\)**: error bound for overlap computation.
- **\(\text{lim}\)**: maximum number of integration terms (Davies, 1980).

Value

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

Author(s)

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

References


pdplot

See Also

MixSim, pdplot, 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)

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

---

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.
- **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)
RandIndex

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)
```

<table>
<thead>
<tr>
<th>RandIndex</th>
<th>Rand’s Index</th>
</tr>
</thead>
</table>

Description

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

Usage

```r
RandIndex(id1, id2)
```

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.
simdataset

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

References

See Also
MixSim, pdplot, simdataset, ClassProp, and VarInf.

Examples

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

---

**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

```r
## 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)

---

**VarInf**

*Variation of Information*

**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

*Topic cluster
  ClassProp, 3
  MixSim, 4
  overlap, 6
  pdplot, 7
  perms, 8
  RandIndex, 10
  simdataset, 11
  VarInf, 13
*Topic datagen
  MixSim, 4
  simdataset, 11
*Topic hplot
  pdplot, 7
*Topic programming
  print.object, 9
  R00_MixSim-package (MixSim-package), 2

ClassProp, 3

MixSim, 4, 10
MixSim-package, 2

overlap, 6

pdplot, 7
perms, 8
print.MixSim(print.object), 9
print.object, 9

RandIndex, 10

simdataset, 11
summary.MixSim(print.object), 9

VarInf, 13