Package ‘HDclassif’

February 19, 2015

Encoding UTF-8
Type Package
Title High Dimensional Supervised Classification and Clustering
Version 1.2.3
Depends graphics, stats, MASS
Date 2015-02-14
Author Laurent Berge, Charles Bouveyron and Stephane Girard
Maintainer Laurent Berge <laurent.berge@u-bordeaux.fr>
Description Discriminant analysis and data clustering methods for high dimensional data, based on the assumption that high-dimensional data live in different subspaces with low dimensionality proposing a new parametrization of the Gaussian mixture model which combines the ideas of dimension reduction and constraints on the model.
License GPL-2
LazyLoad yes
ZipData no
NeedsCompilation no
Repository CRAN
Date/Publication 2015-02-16 06:48:25

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDclassif-package</td>
<td>2</td>
</tr>
<tr>
<td>Crabs</td>
<td>3</td>
</tr>
<tr>
<td>demo_hdce</td>
<td>4</td>
</tr>
<tr>
<td>HDDA</td>
<td>4</td>
</tr>
<tr>
<td>HDDC</td>
<td>8</td>
</tr>
<tr>
<td>plot.hdc</td>
<td>13</td>
</tr>
<tr>
<td>predict.hdce</td>
<td>14</td>
</tr>
<tr>
<td>simuldata</td>
<td>16</td>
</tr>
<tr>
<td>wine</td>
<td>17</td>
</tr>
</tbody>
</table>

Index 19
Description

Discriminant analysis and data clustering methods for high dimensional data, based on the assumption that high-dimensional data live in different subspaces with low dimensionality, proposing a new parametrization of the Gaussian mixture model which combines the ideas of dimension reduction and constraints on the model.

Details

Package: HDclassif
Type: Package
Version: 1.2.3
Date: 2013-02-14
License: GPL-2
LazyLoad: yes

This package is used to make efficient supervised and unsupervised classification with high dimensional data. The supervised method uses the *hdda* function to get the data parameters and the *predict* function to realise the class prediction of a dataset. The unsupervised method is implemented in the *hddc* function, and once the parameters are estimated, the *predict* gives the class prediction of other datasets. The method used in the *hddc* is based on the Expectation - Maximisation algorithm.

Author(s)

Laurent Berge, Charles Bouveyron and Stephane Girard

Maintainer: Laurent Berge <laurent.berge at u-bordeaux.fr>

References


**Description**

The Crabs data frame has 200 rows and 6 columns, describing 5 morphological measurements on 50 crabs each of two colour forms and both sexes, of the species *Leptograpsus Variegatus* collected at Fremantle, W. Australia.

**Usage**

```r
data(Crabs)
```

**Format**

A data frame with 200 observations on the following 6 variables.

- **class** Type of the crabs: the first character represents the species - "B" or "O" for blue or orange-, the second represents the sex -"M" or "F" for male or female-.
- **FL** Frontal lob size (mm).
- **RW** Rear width (mm).
- **CL** Carapace length (mm).
- **CW** Carapace width (mm).
- **BD** Body depth (mm).

**Details**

This dataset can also be found in the MASS package, the unique difference is the class vector which is easier to use here.

**Source**


**References**

**hdda**

**High Dimensional Discriminant Analysis**

**Description**

HDDA is a model-based discriminant analysis method assuming each class of the dataset live in a proper Gaussian subspace which is much smaller than the original one, the hdda.learn function calculates the parameters of each subspace in order to predict the class of new observation of this kind.

**Usage**

```r
hdda(data, cls, model='AkjBkQkDk', graph=FALSE, d="Cattell", threshold=0.2, com_dim=NULL, show=TRUE, scaling=FALSE, cv.dim=1:10, cv.threshold=c(.001, .005, .05, 1:9*0.1), cv.vfold=10, LOO=FALSE, noise.ctrl=1e-8)
```
**Arguments**

- **data**: A matrix or a data frame of observations, assuming the rows are the observations and the columns the variables. Note that NAs are not allowed.
- **cls**: The vector of the class of each observations, its type can be numeric or string.
- **model**: The models used with the HDDA method are the following: “AkjBkQkDk”, “AkBkQkDk”, “ABkQkDk”, “AkBQkDk”, “AkjBkQkD”, “AkBkQkD”, “ABkQkD”, “AkjBQkD”, “AkBQkD”, “ABQkD”, “AjBQD”, “ABQD” and “all”. Note that the model names are not case sensitive. See details for more informations.
- **graph**: It is for comparison sake only, when several estimations are run at the same time (either when using several models, or when using cross-validation to select the best dimension/threshold). If graph = TRUE, the plot of the results of all estimations is displayed. Default is FALSE.
- **d**: Either “Cattell” (default), “BIC” or “CV”. See details for more information.
- **threshold**: A float strictly within 0 and 1. It is the threshold used in the Cattell’s Scree-Test.
- **com_dim**: It is used only for common dimensions models. The user can give the common dimension he wants. If used, it must be an integer. Its default is set to NULL.
- **show**: Use show=FALSE to settle off the informations that may be printed.
- **scaling**: Logical: whether to scale the dataset (mean=0 and standard-deviation=1 for each variable) or not. By default the data is not scaled.
- **cv.dim**: A vector of integers. Only when d=“CV”. Gives the dimensions for which the CV is to be done. Note that if some dimensions are greater than what it is possible to have, those are taken off.
- **cv.threshold**: A vector of floats strictly within 0 and 1. Only when d=“CV”. Gives the thresholds for which the CV is to be done.
- **cv.vfold**: An integer. Only when d=“CV”. It gives the number of different subsamples in which the dataset is split. If “cv.vfold” is greater than the number of observations, then the program equalize them.
- **LOO**: If TRUE, it returns results (classes and posterior probabilities) for leave-one-out cross-validation.
- **noise.ctrl**: This parameter avoids to have a too low value of the ‘noise’ parameter b. It garantees that the dimension selection process do not select too many dimensions (which leads to a potential too low value of the noise parameter b). When selecting the intrinsic dimensions using Cattell’s scree-test or BIC, the function doesn’t use the eigenvalues inferior to noise.ctrl, so that the intrinsic dimensions selected can’t be higher or equal to the order of these eigenvalues.

**Details**

Some information on the signification of the model names:

- **Akj are the parameters of the classes subspaces**:  
  - if Akj: each class has its parameters and there is one parameter for each dimension  
  - if Ak: the classes have different parameters but there is only one per class
• if Aj: all the classes have the same parameters for each dimension (it’s a particular case with a common orientation matrix)
• if A: all classes have the same one parameter

Bk are the noises of the classes subspaces:
• if Bk: each class has its proper noise
• if B: all classes have the same noise

Qk is the orientation matrix of each class:
• if Qk: all classes have its proper orientation matrix
• if Q: all classes have the same orientation matrix

Dk is the intrinsic dimension of each class:
• if Dk: the dimensions are free and proper to each class
• if D: the dimension is common to all classes

The model “all” will compute all the models, give their BIC and keep the model with the highest BIC value. Instead of writing the model names, they can also be specified using an integer. 1 represents the most general model (“AkjBkQkDk”) while 14 is the most constrained (“ABQD”), the others number/name matching are given below. Note also that several models can be run at once, by using a vector of models (e.g. model = c("AKBQKD","AKJBQKD","ABQD") is equivalent to model = c(8,4,13); to run the 6 first models, use model=1:6). If all the models are to be run, model="all" is faster than model=1:14.

| AkjBkQkDk  | AkjBkQkD  | 7 |
| AkBkQkDk  | AkBkQkD  | 8 |
| ABkQkDk  | ABkQkD  | 9 |
| AkjBQkDk  | AkjBQkD  | 10 |
| AkBQkDk  | AkBQkD  | 11 |
| ABQkDk  | ABQkD  | 12 |
| AjBQD  | 13 ABQD  | 14 |

The parameter \(d\), is used to select the intrinsic dimensions of the subclasses. Here are his definitions:

• “Cattell”: The Cattell’s scree-test is used to gather the intrinsic dimension of each class. If the model is of common dimension (models 7 to 14), the scree-test is done on the covariance matrix of the whole dataset.

• “BIC”: The intrinsic dimensions are selected with the BIC criterion. See Bouveyron et al. (2010) for a discussion of this topic. For common dimension models, the procedure is done on the covariance matrix of the whole dataset.

• “CV”: A V-fold cross-validation (CV) can be done in order to select the best threshold (for all models) or the best common dimensions (models 7 to 14). The V-fold cross-validation is done for each dimension (respectively threshold) in the argument “cv.dim” (resp. “cv.threshold”), then the dimension (resp. threshold) that gives the best good classification rate is kept. The dataset is split in “cv.vfold” (default is 10) random subsamples, then CV is done for each sample: each of them is used as validation data while the remaining data is used as training data. For sure, if “cv.vfold” equals the number of observations, then this CV is equivalent to a leave-one-out.

• Note that "Cattell" (resp. "BIC") can be abreviated to "C" (resp. "B") and that this argument is not case sensitive.
Value

hdda returns an ‘hdc’ object; it’s a list containing:

- **model**  The name of the model.
- **k**  The number of classes.
- **d**  The dimensions of each class.
- **a**  The parameters of each class subspace.
- **b**  The noise of each class subspace.
- **mu**  The mean of each variable for each class.
- **prop**  The proportion of each class.
- **ev**  The eigen values of the var/covar matrix.
- **Q**  The orthogonal matrix of orientation of each class.
- **kname**  The name of each class.
- **BIC**  The BIC value of the model used.
- **scaling**  The centers and the standard deviation of the original dataset.

Author(s)

Laurent Berge, Charles Bouveyron and Stephane Girard

References


See Also

hddc, predict.hdc.

Examples

```r
#example 1:
data<-simuldata(1000, 1000, 50, K=5)
X <- data$X
clx <- data$clx
Y <- data$Y
cly <- data$cly
#we get the HDDA parameters:
prms1 <- hdda(X, clx)

c11 <- predict(prms1, Y, cly)
```
#the class vector of Y estimated with HDDA:
c11$class

#another model is used:
prms1 <- hdda(X, clx, model=12)
#model=12 is equivalent to model="ABQkD"
c11 <- predict(prms1, Y, cly)

#example 2:
data(wine)
a <- wine[,-1]
z <- wine[,1]
prms2 <- hdda(a, z, model='all', scaling=TRUE, d="B", graph=TRUE)
c12 <- predict(prms2, a, z)

#getting the best dimension
#using a common dimension model
#we do LOO-CV using cv.vfold=nrow(a)
prms3 <- hdda(a, z, model="akjbqkqd", d="CV", cv.vfold=nrow(a), scaling=TRUE, graph=TRUE)
c13 <- predict(prms3, a, z)

---

**HDCC**

*High Dimensional Data Clustering*

**Description**

HDCC is a model-based clustering method. It is based on the Gaussian Mixture Model and on the idea that the data lives in subspaces with a lower dimension than the dimension of the original space. It uses the Expectation - Maximisation algorithm to estimate the parameters of the model.

**Usage**

```r
hddc(data, K=1:10, model=c("Akjbqkd"), threshold=0.2,
     com_dim=NULL, itermax=60, eps=1e-3, graph=FALSE,
     algo='EM', d="Cattell", init='kmeans', show=TRUE,
     mini.nb=c(5,10), scaling=FALSE, min.individuals=2, noise.ctrl=1e-8, ...)
```

**Arguments**

- **data**
  - A matrix or a data frame of observations, assuming the rows are the observations and the columns the variables. Note that NAs are not allowed.

- **K**
  - A vector of integers specifying the number of clusters for which the BIC and the parameters are to be calculated; the function keeps the parameters which maximises the BIC. Note that the length of the vector K can’t be larger than 20. Default is 1:10.
model A character string vector, or an integer vector indicating the models to be used. The available models are: "AkjBkQkDk" (default), "AkBkQkDk", "ABkQkDk", "AkjBQkDk", "AkBQkDk", "ABQkDk", "AkjBkQkD", "AkBkQkD", "ABkQkD", "AkjBQkD", "AkBQkD", "ABQkD", "AjqBQD", "AbQkD", "AbQkD", "AjBQD", "ABQD". It is not case sensitive and integers can be used instead of names, see details for more information. Several models can be used, if it is, only the results of the one which maximizes the BIC criterion is kept. To run all models, use model="ALL".

threshold A float strictly within 0 and 1. It is the threshold used in the Cattell’s Scree-Test.

com_dim It is used only for common dimensions models. The user can give the common dimension he wants. If used, it must be an integer. Its default is set to NULL.

itermax The maximum number of iterations allowed. The default is 60.

eps A positive double. It is the stopping criterion: the algorithm stops when the difference between two successive Log Likelihoods is lower than 'eps'.

graph It is for comparison sake only, when several estimations are run at the same time (either when using several models, or when using cross-validation to select the best dimension/threshold). If graph = TRUE, the plot of the results of all estimations is displayed. Default is FALSE.

algo A character string indicating the algorithm to be used. The available algorithms are the Expectation-Maximisation ("EM"), the Classification E-M ("CEM") and the Stochastic E-M ("SEM"). The default algorithm is the "EM".

d Either “Cattell” (default) or “BIC”. See details for more information.

init A character string or a vector of clusters. It is the way to initialize the E-M algorithm. There are five ways of initialization: “kmeans” (default), “param”, “random” or “mini-em”. See details for more information. It can also be directly initialized with a vector containing the prior classes of the observations.

show Use show = FALSE to settle off the informations that may be printed.

mini.nb A vector of integers of length two. This parameter is used in the "mini-em" initialization. The first integer sets how many times the algorithm is repeated; the second sets the maximum number of iterations the algorithm will do each time. For example, if init="mini-em" and mini.nb=c(5,10), the algorithm will be launched 5 times, doing each time 10 iterations; finally the algorithm will begin with the initialization that maximizes the log-likelihood.

scaling Logical: whether to scale the dataset (mean=0 and standard-error=1 for each variable) or not. By default the data is not scaled.

min.individuals This parameter is used to control for the minimum population of a class. If the population of a class becomes strictly inferior to 'min.individuals' then the algorithm stops and gives the message: 'pop<min.indiv.'. Here the meaning of "population of a class" is the sum of its posterior probabilities. The value of 'min.individuals' cannot be lower than 2.

noise.ctrl This parameter avoids to have a too low value of the 'noise' parameter b. It guarantees that the dimension selection process do not select too many dimensions (which leads to a potential too low value of the noise parameter b). When selecting the intrinsic dimensions using Cattell’s scree-test or BIC, the function doesn’t use the eigenvalues inferior to noise.ctrl, so that the intrinsic dimensions selected can’t be higher or equal to the order of these eigenvalues.
Any arguments that can be used by the function "kmeans" when this initialization is selected (namely: algorithm; nstart; iter.max).

Details

Some information on the signification of the model names:

Akj are the parameters of the classes subspaces:  • if Akj: each class has its parameters and there is one parameter for each dimension
  • if Ak: the classes have different parameters but there is only one per class
  • if Aj: all the classes have the same parameters for each dimension (it’s a particular case with a common orientation matrix)
  • if A: all classes have the same one parameter

Bk are the noises of the classes subspaces:  • If Bk: each class has its proper noise
  • if B: all classes have the same noise

Qk is the orientation matrix of each class:  • if Qk: all classes have its proper orientation matrix
  • if Q: all classes have the same orientation matrix

Dk is the intrinsic dimension of each class:  • if Dk: the dimensions are free and proper to each class
  • if D: the dimension is common to all classes

The model “ALL” will compute all the models, give their BIC and keep the model with the highest BIC value. Instead of writing the model names, they can also be specified using an integer. 1 represents the most general model (“AkjBkQkDk”) while 14 is the most constrained (“ABQD”), the others number/name matching are given below. Note also that several models can be run at once, by using a vector of models (e.g. model = c(“AKBQKD”,“AKJBQKDK”,”AJBQD”) is equivalent to model = c(8,4,13); to run the 6 first models, use model=1:6). If all the models are to be run, model=“all” is faster than model=1:14.

<table>
<thead>
<tr>
<th>AkjBkQkDk</th>
<th>AkjBkQkD</th>
<th>AkBkQkD</th>
<th>ABkQkD</th>
<th>AkjBQkD</th>
<th>AkBQkD</th>
<th>ABQkD</th>
<th>AjbQD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>11</td>
<td>6</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

The parameter d, is used to select the intrinsic dimensions of the subclasses. Here are his definitions:

• “Cattell”: The Cattell’s scree-test is used to gather the intrinsic dimension of each class. If the model is of common dimension (models 7 to 14), the scree-test is done on the covariance matrix of the whole dataset.

• “BIC”: The intrinsic dimensions are selected with the BIC criterion. See Bouveyron et al. (2010) for a discussion of this topic. For common dimension models, the procedure is done on the covariance matrix of the whole dataset.
• Note that "Cattell" (resp. "BIC") can be abbreviated to "C" (resp. "B") and that this argument is not case sensitive.

The different initializations are:

“param”: it is initialized with the parameters, the means being generated by a multivariate normal distribution and the covariance matrix being common to the whole sample

“mini-em”: it is an initialization strategy, the classes are randomly initialized and the E-M algorithm makes several iterations, this action is repeated a few times (the default is 5 iterations and 10 times), at the end, the initialization chosen is the one which maximizes the log-likelihood (see mini.nb for more information about its parametrization)

“random”: the classes are randomly given using a multinomial distribution

“kmeans”: the classes are initialized using the kmeans function (with: algorithm="Hartigan-Wong"; nstart=4; iter.max=50); note that the user can use his own arguments for kmeans using the dot-dot-dot argument

A prior class vector: It can also be directly initialized with a vector containing the prior classes of the observations

Value

hddc returns an 'hdc' object; it's a list containing:

- model: The name of the model.
- K: The number of classes.
- d: The dimensions of each class.
- a: The parameters of each class subspace.
- b: The noise of each class subspace.
- mu: The mean of each variable for each class.
- prop: The proportion of each class.
- ev: The eigen values of the var/covar matrix.
- Q: The orthogonal matrix of orientation of each class.
- kname: The name of each class.
- BIC: The BIC value of the model used.
- scaling: The centers and the standard deviation of the original dataset.
- likelihood: The Log Likelihood for each iteration.
- class: The class vector obtained by the clustering.
- posterior: The matrix of the probabilities to belong to a class for each observation and each class.

Author(s)

Laurent Berge, Charles Bouveyron and Stephane Girard
References

tional Statistics and Data Analysis*, vol. 52 (1), pp. 502–519

tering and Discriminant Analysis of High-Dimensional Data”, *Journal of Statistical Software*, 46(6),
1–29, url: http://www.jstatsoft.org/v46/i06/

See Also

hdda, predict.hdc, plot.hdc.

Examples

```r
# Example 1:
data <- simuldata(1000, 1000, 50)
X <- data$X
clx <- data$clx
Y <- data$Y
cly <- data$cly

# clustering of the simulated dataset:
prms1 <- hddc(X, K=3, algo="CEM", init='param')

# class vector obtained by the clustering:
prms1$class

# We can look at the adjusted rand index to assess the goodness of fit
res1 <- predict(prms1, X, clx)
res2 <- predict(prms1, Y)
# the class predicted using hddc parameters on the test dataset:
res2$class

# Example 2:
data(Crabs)

# clustering of the Crabs dataset:
prms3 <- hddc(Crabs[-1], K=4, algo="EM", init='mini-em')
res3 <- predict(prms3, Crabs[-1], Crabs[,1])

# another example using the Crabs dataset
prms4 <- hddc(Crabs[-1], K=1:8, graph=TRUE, model=c(1,2,7,9))
# model=c(1,2,7,9) is equivalent to model=c("AKJBKQKD","AKBKQKDK","AKJBKQKD","ABKQKD")
res4 <- predict(prms4, Crabs[-1], Crabs[,1])
```
Description

This function plots Cattell’s scree-test or the BIC selection, using parameters coming from \texttt{hdda} or \texttt{hddc} functions.

Usage

```r
## S3 method for class 'hdc'
plot(x, method = NULL, threshold = NULL, noise.ctrl=1e-8, ...)
```

Arguments

- \texttt{x} A 'hdc' class object obtained using \texttt{hdda} or \texttt{hddc} methods.
- \texttt{method} The method used to select the intrinsic dimension. It can be "BIC" or "Cattell". By default it takes the method used when obtaining the parameters using \texttt{hdda} or \texttt{hddc}. Note that "Cattell" (resp. "BIC") can be abbreviated to "C" (resp. "B") and that this argument is not case sensitive.
- \texttt{threshold} The threshold used in Cattell’s Scree-Test. By default it takes the threshold in the argument \texttt{x}, if none, the default value of the threshold is 0.2.
- \texttt{noise.ctrl} This parameter avoids to have a too low value of the 'noise' parameter \(b\). It guarantees that the dimension selection process do not select too many dimensions (which leads to a potential too low value of the noise parameter \(b\)). When selecting the intrinsic dimensions using Cattell’s scree-test or BIC, the function doesn’t use the eigenvalues inferior to \texttt{noise.ctrl}, so that the intrinsic dimensions selected can’t be higher or equal to the order of these eigenvalues.

Value

- \textbf{If method = "Cattell"} The plot of the eigen values and of the sequential differences of the eigen values. The dimension to retain is the one before the last fall of the eigenvalues’ differences below the threshold.
- \textbf{If method = "BIC"} The BIC related to the dimension for each class. It stops after the first fall of the BIC.

Author(s)

Laurent Berge, Charles Bouveyron and Stephane Girard
References


See Also

`hdda`, `hddc`, `predict.hdc`.

Examples

```r
# Example 1:
data(wine)
a <- wine[, -1]
z <- wine[, 1]
prms1 <- hdda(a, z, model="AkBkQkDk", scaling=TRUE, d="B")
# the plot related to the selection that has been done: BIC
plot(prms1)

# it shows the plot of Cattell's scree-test, with a threshold of .3
plot(prms1,"Cattell",.3)

prms2 <- hdda(a, z, model="AkBkQkDk", scaling=TRUE, d="c")
# the plot related to the selection that has been done: Cattell's scree-test
plot(prms2)
# the plot of the BIC
plot(prms2,"b")
```

**predict.hdc**

*Prediction method for 'hdc' class objects.*

**Description**

This function computes the class prediction of a dataset with respect to the model-based supervised and unsupervised classification methods `hdda` and `hddc`.

**Usage**

```r
## S3 method for class 'hdc'
predict(object, data, cls = NULL, ...)
```
predict.hdc

Arguments

- **object**: An 'hdc' class object obtained by using hdda or hddc function.
- **data**: A matrix or a data frame of observations, assuming the rows are the observations and the columns the variables. Note that NAs are not allowed.
- **cls**: It is optional and used to be compared to the predicted classes, default is NULL.

Value

- **class**: vector of the predicted class.
- **prob**: The matrix of the probabilities to belong to a class for each observation and each class.

If the initial class vector is given to the argument ‘cls’ then the adjusted rand index (ARI) is also returned. Also the following object is returned:

- **ARI**: The confusion matrix of the classification.

Author(s)

Laurent Berge, Charles Bouveyron and Stephane Girard

References


See Also

hdda, hddc.

Examples

```r
# Example 1:
data <- simuldata(1000, 1000, 50)
X <- data$X
clx <- data$clx
Y <- data$Y
cly <- data$cly

#clustering of the gaussian dataset:
prms1 <- hddc(X, K=3, algo="CEM", init='param')

#class vector obtained by the clustering:
```
prms1$class

# only to see the good classification rate and the confusion matrix:
res1 <- predict(prms1, X, clx)
res2 <- predict(prms1, Y)

# the class predicted using hddc parameters on the test dataset:
res2$class

# Example 2:
data(Crabs)
# clustering of the Crabs dataset:
prms3 <- hddc(Crabs[,,-1], K=4, algo="EM", init='kmeans')
res3 <- predict(prms3, Crabs[,,-1], Crabs[,1])

# the confusion matrix:
res3$confusion

---

**simuldata**

*Gaussian Data Generation*

**Description**

This function generates two datasets according to the model \([AkBkQkDk]\) of the HDDA gaussian mixture model paramatrisation (see ref.).

**Usage**

`simuldata(nlearn, ntest, p, K = 3, prop = NULL, d = NULL, a = NULL, b = NULL)`

**Arguments**

- `nlearn` The size of the learning dataset to be generated.
- `ntest` The size of the testing dataset to be generated.
- `p` The number of variables.
- `K` The number of classes.
- `prop` The proportion of each class.
- `d` The dimension of the intrinsic subspace of each class.
- `a` The value of the main parameter of each class.
- `b` The noise of each class.
**Value**

- $X$ The learning dataset.
- $\text{clx}$ The class vector of the learning dataset.
- $Y$ The test dataset.
- $\text{cly}$ The class vector of the test dataset.
- $\text{prms}$ The principal parameters used to generate the datasets.

**Author(s)**

Laurent Berge, Charles Bouveyron and Stephane Girard

**References**


**See Also**

hdda, hddc.

**Examples**

```r
data <- simuldata(500, 1000, 50, K=5, prop=c(0.2,0.25,0.25,0.15,0.15))
X <- data$X
clx <- data$clx
f <- hdda(X, clx)
Y <- data$Y
cly <- data$cly
e <- predict(f, Y, cly)
```

---

**wine**

*Wine dataset*

**Description**

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

**Usage**

```r
data(wine)
```
Format

A data frame with 178 observations on the following 14 variables:

- **class**: The class vector, the three different cultivars of wine are represented by the three integers: 1 to 3.
- **V1**: Alcohol
- **V2**: Malic acid
- **V3**: Ash
- **V4**: Alcalinity of ash
- **V5**: Magnesium
- **V6**: Total phenols
- **V7**: Flavanoids
- **V8**: Nonflavanoid phenols
- **V9**: Proanthocyanins
- **V10**: Color intensity
- **V11**: Hue
- **V12**: OD280/OD315 of diluted wines
- **V13**: Proline

Source

This dataset is from the UCI machine learning repository, provided here: http://archive.ics.uci.edu/ml/datasets/Wine.

Examples

data(wine)
Index

*Topic **cattell**
plot.hdc, 13

*Topic **classification**
HDDA, 4

*Topic **clustering**
HDDC, 8
plot.hdc, 13
predict.hdc, 14

*Topic **datasets**
Crabs, 3
wine, 17

*Topic **demo**
demo_hddc, 4

*Topic **gaussian**
simuldata, 16

*Topic **generation**
simuldata, 16

*Topic **hdda**
HDDA, 4
plot.hdc, 13
predict.hdc, 14

*Topic **hddc**
HDDC, 8
plot.hdc, 13
predict.hdc, 14

*Topic **package**
Hdclassif-package, 2

*Topic **predict**
HDDA, 4

Crabs, 3
demo_hddc, 4

Hdclassif (Hdclassif-package), 2
Hdclassif-package, 2
HDDA, 4
hdda, 12, 14, 15, 17
hdda (HDDA), 4
HDDC, 8

plot.hdc, 12, 13
predict.hdc, 7, 12, 14, 14
simuldata, 16
wine, 17

19