Package ‘lpda’

July 16, 2025

Type Package

Title Linear Programming Discriminant Analysis

Version 1.2.1

Description Classification method obtained through linear programming.
It is advantageous with respect to the classical developments when the distribution of the variables involved is unknown or when the number of variables is much greater than the number of individuals.
Mathematical details behind the method are published in Nueda, et al. (2022) ``LPDA: A new classification method based on linear programming''.
<doi:10.1371/journal.pone.0270403>.

Depends R (>= 3.5.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rglpk, multiway

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author María José Nueda [aut, cre]

Maintainer María José Nueda <`mj.nueda@ua.es`>

Repository CRAN

Date/Publication 2025-07-16 09:50:06 UTC

Contents

`lpda` *Computing discriminating hyperplane for two groups*

Description

This function computes a discriminating hyperplane for two groups with original data (calling lpda.fit) or with principal components (calling lpda.pca)

Usage

```
lpda(data, group, scale = FALSE, pca = FALSE, PC = 2, Variability = NULL,
                 f1 = NULL, f2 = NULL)
## S3 method for class 'lpda'
print(x, ...)
```

Arguments

`data`	Matrix containing data. Individuals in rows and variables in columns
`group`	Vector with the variable group
`scale`	Logical indicating if it is required standardize data. When pca=TRUE data is always scaled.
`pca`	Logical indicating if Principal Components Analysis is required
`PC`	Number of Principal Components (PC) for PCA. By default it is 2. When the number of PC is not decided, it can be determined choosing the desired proportion of explained variability (Variability parameter).
`Variability`	Parameter for Principal Components (PC) selection. This is the minimum desired proportion of variability explained for the PC of the variables. The analysis is always done with a minimum of 2 PCs. If it is NULL the PCA will be computed with PC parameter.

`f1`	Vector with weights for individuals of the first group. If NULL they are equally weighted.
`f2`	Vector with weights for individuals of the second group. If NULL they are equally weighted.
`x`	An object of class "`lpda`", a result of a call to `lpda`.
`...`	Other arguments passed.

Value

`lpda` returns an object of class "`lpda`".

`coef`	Hyperplane coefficients
`data`	Input data matrix
`group`	Input group vector
`scale`	Input scale argument
`pca`	Input pca argument
`loadings`	Principal Components loadings. Showed when pca = TRUE
`scores`	Principal Components scores. Showed when pca = TRUE
`var.exp`	A matrix containing the explained variance for each component and the cumulative variance. Showed when pca = TRUE
`PCs`	Number of Principal Components in the analysis. Showed when pca = TRUE

The functions `predict` and `plot` can be used to obtain the predicted classes and a plot in two dimensions with the distances to the computed hyperplane for the two classes.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

References

Nueda MJ, Gandía C, Molina MD (2022) LPDA: A new classification method based on linear programming. PLoS ONE 17(7): e0270403. <https://doi.org/10.1371/journal.pone.0270403>

See Also

`lpda.pca`, `lpda.fit`

Examples

```
######### palmdates example in lpda package:
data(palmdates)
group = as.factor( c(rep("Spanish",11),rep("Foreign",10)) )

# with concentration data:
model = lpda(data = palmdates$conc, group = group )
summary(model)
```

```
predict(model)
plot(model, main = "Palmdates example")

model.pca = lpda(data = palmdates$conc, group = group, pca=TRUE, PC = 2)
plot(model.pca, PCscores = TRUE, main = "Palmdates example")

# with spectra data
model.pca = lpda(data = palmdates$spectra, group = group, pca=TRUE, Variability = 0.9)
model.pca$PCs # 4 PCs to explain 90% of the variability
plot(model.pca, PCscores = TRUE, main = "Spectra palmdates")
```

`lpda.3D` *Classification with lpda for 3way array data*

Description

This function applies lpda methodology to classify individuals in two or more groups with original data (by applying lpda through the third dimension) or by applying lpda to the parafac scores.

Usage

```
lpda.3D(data, group, scale = FALSE,  pfac = FALSE, nfac = 2,
                     nstart = 10, seed=2, f1 = NULL, f2 = NULL)
## S3 method for class 'lpda.3D'
print(x, ...)
```

Arguments

`data`	Array containing data. Individuals in the first mode, variables in the second mode and third mode with time or similar.
`group`	Vector with the variable group.
`scale`	Logical indicating if it is required standardize data.
`pfac`	Logical indicating if Parafac Analysis is required
`nfac`	Number of factors for Parafac Analysis.
`nstart`	Number of random starts for multiway analysis.
`seed`	A single value to reproduce same results in multiway methods.
`f1`	Vector with weights for individuals of the first group. If NULL they are equally weighted.
`f2`	Vector with weights for individuals of the second group. If NULL they are equally weighted.
`x`	An object of class "`lpda.3D`", a result of a call to `lpda.3D`.
`...`	Other arguments passed.

Value

`lpda.3D` returns an object of class "`lpda.3D`".

`MOD`	When `pfac=FALSE`, MOD is a list of `lpda` objects obtained from the application of lpda to each slide matrix of the third node. When `pfac=TRUE`, MOD is a list of two elements: (1) `mod.pfac` the parafac model obtained with Multiway package and (2) the `lpda` object, obtained from the application of lpda to the parafac scores.
`data`	Input array data
`group`	Input group vector
`pfac`	Input pfac argument

The functions `predict` and `plot` can be used to obtain the predicted classes and a plot in two dimensions with the distances to the computed hyperplane for the two classes.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

References

Nueda MJ, Gandía C, Molina MD (2022) LPDA: A new classification method based on linear programming. PLoS ONE 17(7): e0270403. <https://doi.org/10.1371/journal.pone.0270403>

See Also

`lpda`, `parafac`

Examples

```
### RNAseq is a 3-dimensional array
  data(RNAseq)
  group = as.factor(rep(c("G1","G2"), each = 10))

## Strategy 1
  model3D = lpda.3D(RNAseq, group)
  summary(model3D)
  predict(model3D)
  plot(model3D, mfrow=c(2,2))

## Strategy 2: with parafac
  model3Ds2 = lpda.3D(RNAseq, group, pfac=TRUE, nfac=2)
  model3Ds2$MOD$mod.pfac$Rsq
  predict(model3Ds2)
  summary(model3Ds2)
  plot(model3Ds2, pfacscores=FALSE, main="Parafac Model", mfrow=c(1,1))
  plot(model3Ds2, pfacscores=TRUE, cex=1.5, main="Parafac components")
  legend("bottomright", levels(group), col=c(2,3), pch=20)
```

`lpda.fit` *lpda.fit computes the discriminating hyperplane for two groups*

Description

`lpda.fit` computes the discriminating hyperplane for two groups, giving as a result the coefficients of the hyperplane.

Usage

```
lpda.fit(data, group, f1 = NULL, f2 = NULL)
```

Arguments

`data`	Matrix containing data. Individuals in rows and variables in columns
`group`	Vector with the variable group
`f1`	Vector with weights for individuals of the first group
`f2`	Vector with weights for individuals of the second group

Value

`coef`	Hyperplane coefficients

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

References

Nueda MJ, Gandía C, Molina MD (2022) LPDA: A new classification method based on linear programming. PLoS ONE 17(7): e0270403. <https://doi.org/10.1371/journal.pone.0270403>

See Also

`lpda`

`lpda.pca` *lpda.pca computes a PCA to the original data and selects the desired PCs when Variability is supplied*

Description

`lpda.pca` computes the discriminating hyperplane for two groups with Principal Components (PC)

Usage

```
lpda.pca(data, group, PC = 2, Variability = NULL)
```

Arguments

`data`	Matrix containing data. Individuals in rows and variables in columns
`group`	Vector with the variable group
`PC`	Number of Principal Components (PC) for PCA. By default it is 2. When the number of PC is not decided, it can be determined choosing the desired proportion of explained variability (Variability parameter).
`Variability`	Parameter for Principal Components (PC) selection. This is the minimum desired proportion of variability explained for the PC of the variables. The analysis is always done with a minimum of 2 PCs. If it is NULL the PCA will be computed with PC parameter.

Value

`loadings`	Principal Components loadings.
`scores`	Principal Components scores.
`var.exp`	A matrix containing the explained variance for each component and the cumulative variance.
`PCs`	Number of Principal Components in the analysis.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

References

Nueda MJ, Gandía C, Molina MD (2022) LPDA: A new classification method based on linear programming. PLoS ONE 17(7): e0270403. <https://doi.org/10.1371/journal.pone.0270403>

See Also

`lpda`

`lpdaCV` *Crossvalidation procedure for lpda evaluation*

Description

lpdaCV evaluates the error rate classification with a crossvalidation procedure

Usage

```
lpdaCV(data, group, scale = FALSE, pca = FALSE, PC = 2, Variability = NULL,
                   CV = "ktest", ntest = 10, R = 10, f1 = NULL, f2 = NULL)

## S3 method for class 'lpdaCV'
print(x, ...)
```

Arguments

`data`	Matrix containing data. Individuals in rows and variables in columns
`group`	Vector with the variable group
`scale`	Logical indicating if it is required standardize data.
`pca`	Logical indicating if a reduction of dimension is required
`PC`	Number of Principal Components (PC) for PCA. By default it is 2. When the number of PC is not decided, it can be determined choosing the desired proportion of explained variability (Variability parameter) or choosing the maximum number of errors allowed in the training set (Error.max).
`Variability`	Parameter for Principal Components (PC) selection. This is the desired proportion of variability explained for the PC of the variables.
`CV`	Crossvalidation mode: loo "leave one out" or ktest: that leaves k in the test set.
`ntest`	Number of samples to evaluate in the test-set.
`R`	Number of times that the error is evaluated.
`f1`	Vector with weights for individuals of the first group. If NULL they are equally weighted.
`f2`	Vector with weights for individuals of the second group. If NULL they are equally weighted.
`x`	An object of class "`lpdaCV`", a result of a call to `lpdaCV` or `lpdaCV.3D`.
`...`	Other arguments passed.

Value

`lpdaCV` returns the prediction error rate classification.

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

See Also

lpda

Examples

```
### RNAseq is a 3-dimensional array
  data(RNAseq)
  data = RNAseq[,,3]
  group = as.factor(rep(c("G1","G2"), each = 10))
  lpdaCV(data, group, pca = TRUE, CV = "ktest", ntest = 2)
```

`lpdaCV.3D` *Crossvalidation procedure for lpda3D evaluation*

Description

lpdaCV.3D evaluates the error rate classification with a crossvalidation procedure

Usage

```
lpdaCV.3D(data, group, scale = FALSE, pfac = FALSE, nfac = 2, nstart = 10, seed=2,
                         CV = "ktest", ntest = 10, R = 10, f1 = NULL, f2 = NULL)
```

Arguments

`data`	Array containing data. Individuals in the first mode, variables in the second mode and third mode with time or similar.
`group`	Vector with the variable group.
`scale`	Logical indicating if it is required standardize data.
`pfac`	Logical indicating if Parafac Analysis is required
`nfac`	Number of factors for Parafac Analysis. By default it is 2.
`nstart`	Number of random starts for multiway analysis.
`seed`	A single value to reproduce same results in multiway methods. If NULL the start will be random.
`CV`	Crossvalidation mode: loo "leave one out" or ktest: that leaves k in the test set.
`ntest`	Number of samples to evaluate in the test-set.
`R`	Number of times that the error is evaluated.
`f1`	Vector with weights for individuals of the first group. If NULL they are equally weighted.
`f2`	Vector with weights for individuals of the second group. If NULL they are equally weighted.

Value

`lpda.3D` returns the prediction error rate classification.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

See Also

`lpda.3D`, `lpdaCV`

Examples

```
### RNAseq is a 3-dimensional array

  data(RNAseq)
  group = as.factor(rep(c("G1","G2"), each = 10))
  lpdaCV.3D(RNAseq, group , CV = "ktest", R=5, ntest=5, pfac=TRUE, nfac=c(2,10))
```

`palmdates` *Spectrometry and composition chemical of Spanish and Arabian palm dates*

Description

A data set with scores of 21 dates on spectrometry and concentration measurements of the substances that better define the quality of the dates: fibre, sorbitol, fructose, glucose and myo-inositol. The first 11 dates are Spanish (from Elche, Alicante) and the last 10 are from other countries, mainly Arabian.

Usage

```
palmdates
```

Format

A data frame with 2 elements:

conc a data frame with 5 columns: fibre, sorbitol, fructose, glucose and myo-inositol.

spectra a data frame with 2050 columns.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

References

Abdrabo, S.S., Gras, L., Grindlay, G. and Mora, J. (2021) Evaluation of Fourier Transform-Raman Spectroscopy for palm dates characterization. Journal of food composition and analysis. Submitted.

`PCA` *Principal Component Analysis*

Description

Computes a Principal Component Analysis when p>n and when p<=n.

Usage

```
PCA(X)
```

Arguments

`X`	Matrix or data.frame with variables in columns and observations in rows.

Value

`eigen`	A eigen class object with eigenvalues and eigenvectors of the analysis.
`var.exp`	A matrix containing the explained variance for each component and the cumulative variance.
`scores`	Scores of the PCA analysis.
`loadings`	Loadings of the PCA analysis.

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

Examples

```
## Simulate data matrix with 500 variables and 10 observations
datasim = matrix(sample(0:100, 5000, replace = TRUE), nrow = 10)
## PCA
myPCA = PCA(datasim)
## Extracting the variance explained by each principal component
myPCA$var.exp
```

`plot.lpda` *Plot method for lpda classification*

Description

`plot.lpda` is applied to an `lpda` class object. It shows a plot in two dimensions with the distances to the computed hyperplane of each individual coloring each case with the real class.

Usage

```
## S3 method for class 'lpda'
plot(x, PCscores = FALSE, main = NULL, xlab = NULL, ylab = NULL,
                    col= NULL, pch = NULL, lty = NULL,
                    legend.pos = "topright",...)
```

Arguments

`x`	Object of class inheriting from "lpda"
`PCscores`	Logical to show the first 2 PCscores. Only possible when PCA is applied.
`main`	An optional title for the plot.
`xlab`	An optional title for x-axis.
`ylab`	An optional title for y-axis.
`col`	An optional vector with colours for the groups.
`pch`	An integer specifying the symbol to be used in plotting points. When NULL, pch=20.
`lty`	The line type. If it is not specified, lty = 2 for the distances to the hiperplane and lty = 1 for PCs plot
`legend.pos`	The position for the legend. By default it is topright. NULL when no legend is required.
`...`	Other arguments passed.

Value

Two dimensional plot representing the distances to the computed hyperplane of each individual colored with the real class.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

See Also

`lpda`

`plot.lpda.3D` *Plot method for lpda classification*

Description

`plot.lpda.3D` is applied to an `lpda.3D` class object. It shows a plot in two dimensions with the distances to the computed hyperplane of each individual coloring each case with the real class.

Usage

```
## S3 method for class 'lpda.3D'
plot(x, pfacscores = FALSE, main = NULL,
legend.pos = "topright", ...)
```

Arguments

`x`	Object of class inheriting from "lpda"
`pfacscores`	Logical to show the first 2 parafac scores. Only possible when parafac is applied.
`main`	An optional title for the plot.
`legend.pos`	The position for the legend. By default it is topright. NULL when no legend is required.
`...`	Other arguments passed.

Value

Two dimensional plot representing the distances to the computed hyperplane of each individual colored with the real class.

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

See Also

`lpda.3D`, `plot.lpda`

`predict.lpda` *Predict method for lpda classification*

Description

Predict method for lpda classification

Usage

```
## S3 method for class 'lpda'
predict(object, datatest = object$data,...)

## S3 method for class 'predict.lpda'
print(x, ...)
```

Arguments

`object`	Object of class inheriting from "lpda"
`datatest`	An optional data to predict their class. If omitted, the original data is used.
`x`	An object of class "`predict.lpda`", a result of a call to `predict.lpda`.
`...`	Other arguments passed.

Value

`fitted`	Predicted class
`eval`	Evaluation of each individual in the fitted model

Author(s)

Maria Jose Nueda, <`mj.nueda@ua.es`>

See Also

`lpda`

`predict.lpda.3D` *Predict method for lpda.3D classification*

Description

Predict method for lpda.3D classification

Usage

```
## S3 method for class 'lpda.3D'
predict(object, datatest = NULL,...)

## S3 method for class 'predict.lpda.3D'
print(x, ...)
```

Arguments

`object`	Object of class inheriting from "lpda.3D"
`datatest`	An optional data to predict their class. If omitted, the original data is used.
`x`	An object of class "`predict.lpda.3D`", a result of a call to `predict.lpda.3D`.
`...`	Other arguments passed.

Value

`fitted`	Predicted class
`eval`	Evaluation of each individual in all the fitted models

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

See Also

`lpda`

`RNAseq` *Simulated RNA-Seq Time Series 3-dimensional dataset example*

Description

A simulated RNA-Seq dataset example.

Usage

```
RNAseq
```

Format

An array data frame with 20 samples (1st dimension), 600 genes (2nd dimension) and 4 time-points (3rd dimension).

Details

This dataset is a RNA-Seq simulated example. It has been simulated as Negative Binomial distributed and transformed to rpkm (Reads per kilo base per million mapped reads). It contains gene expression from 600 genes measured to 60 samples through 4 time-points. First 10 samples are from first group and the remaining samples from the second one.

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

`stand` *stand center and scale a data matrix*

Description

`stand` center and scale a data matrix

Usage

```
stand(X)
```

Arguments

`X`	a data matrix with individuals in rows and variables in columns

Value

Scaled data matrix

`stand2` *stand2 center and scale a data matrix with the parameters of another one*

Description

`stand2` center and scale a data matrix with the parameters of another one

Usage

```
stand2(X, X2)
```

Arguments

`X`	the data matrix from which mean and standard deviation is computed
`X2`	the data matrix to center and scale

Value

Scaled X2 data matrix

`summary.lpda` *Summarizing lpda classification*

Description

`summary` method for class "lpda"

Usage

```
## S3 method for class 'lpda'
summary(object, datatest = object$data, grouptest=object$group,...)

## S3 method for class 'summary.lpda'
print(x, ...)
```

Arguments

`object`	Object of class inheriting from "lpda"
`datatest`	An optional data to predict their class and compare with real in the confusion matrix. If omitted, the original data is used.
`grouptest`	When datatest is specified, grouptest must also be specified and viceversa.
`x`	An object of class "`summary.lpda`", a result of a call to `summary.lpda`.
`...`	Other arguments passed.

Value

`Confusion.Matrix`	Table of confusion. Predicted classes in rows and real classes in columns, giving the hit (in the diagonal) and misclassification counts (out of the diagonal)

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

See Also

`lpda`

`summary.lpda.3D` *Summarizing lpda.3D classification*

Description

summary method for class "lpda.3D"

Usage

```
## S3 method for class 'lpda.3D'
summary(object, datatest = NULL, grouptest=NULL,...)

## S3 method for class 'summary.lpda.3D'
print(x, ...)
```

Arguments

`object`	Object of class inheriting from "lpda.3D"
`datatest`	An optional data to predict their class and compare with real in the confusion matrix. If omitted, the original data is used.
`grouptest`	When datatest is specified, grouptest must also be specified and viceversa.
`x`	An object of class "`summary.lpda.3D`", a result of a call to `summary.lpda.3D`.
`...`	Other arguments passed.

Value

`Confusion.Matrix`	Table of confusion. Predicted classes in rows and real classes in columns, giving the hit (in the diagonal) and misclassification counts (out of the diagonal)

Author(s)

Maria Jose Nueda, `<mj.nueda@ua.es>`

See Also

`lpda.3D`

Index