# Package 'folda'

September 11, 2024

**Title** Forward Stepwise Discriminant Analysis with Pillai's Trace

**Version** 0.1.0

**Description** A novel forward stepwise discriminant analysis framework
that integrates Pillai's trace with Uncorrelated Linear Discriminant Analysis (ULDA),
providing an improvement over traditional stepwise LDA methods that rely on Wilks' Lambda.
A stand-alone ULDA implementation is also provided, offering a more general solution
than the one available in the 'MASS' package. It automatically handles missing values and
provides visualization tools. For more de-
tails, see Wang (2024) <doi:10.48550/arXiv.2409.03136>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Imports** ggplot2, grDevices, Rcpp, stats

**URL** `https://github.com/Moran79/folda`, `http://iamwangsiyu.com/folda/`

**BugReports** `https://github.com/Moran79/folda/issues`

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**LinkingTo** Rcpp, RcppEigen

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Siyu Wang [aut, cre, cph] (<`https://orcid.org/0009-0005-2098-7089`>)

**Maintainer** Siyu Wang <`iamwangsiyu@gmail.com`>

**Repository** CRAN

**Date/Publication** 2024-09-11 16:40:02 UTC

## Contents

---

`checkPriorAndMisClassCost`
*Check and Normalize Prior Probabilities and Misclassification Costs*

---

## Description

This function verifies and normalizes the provided prior probabilities and misclassification cost matrix for a given response variable. It ensures that the lengths of the prior and the dimensions of the misclassification cost matrix match the number of levels in the response variable. If `prior` or `misClassCost` are not provided, default values are used: the prior is set to the observed frequencies of the response, and the misclassification cost matrix is set to 1 for all misclassifications and 0 for correct classifications.

## Usage

```
checkPriorAndMisClassCost(prior, misClassCost, response)
```

## Arguments

| | |
|---|---|
| `prior` | A numeric vector representing the prior probabilities for each class in the response variable. If `NULL`, the observed frequencies of the response are used as the default prior. |
| `misClassCost` | A square matrix representing the misclassification costs for each pair of classes in the response variable. If `NULL`, a default misclassification matrix is created where all misclassifications have a cost of 1 and correct classifications have a cost of 0. |
| `response` | A factor representing the response variable with multiple classes. |

## Value

A list containing:

| | |
|---|---|
| `prior` | A normalized vector of prior probabilities for each class. |
| `misClassCost` | A square matrix representing the misclassification costs, with rows and columns labeled by the levels of the response variable. |

---

`folda` *Forward Uncorrelated Linear Discriminant Analysis*

---

## Description

This function fits a ULDA (Uncorrelated Linear Discriminant Analysis) model to the provided data, with an option for forward selection of variables based on Pillai's trace or Wilks' Lambda. It can also handle missing values, perform downsampling, and compute the linear discriminant scores and group means for classification. The function returns a fitted ULDA model object.

## Usage

```
folda(
  datX,
  response,
  subsetMethod = c("forward", "all"),
  testStat = c("Pillai", "Wilks"),
  correction = TRUE,
  alpha = 0.1,
  prior = NULL,
  misClassCost = NULL,
  missingMethod = c("medianFlag", "newLevel"),
  downSampling = FALSE,
  kSample = NULL
)
```

## Arguments

| | |
|---|---|
| `datX` | A data frame containing the predictor variables. |
| `response` | A factor representing the response variable with multiple classes. |
| `subsetMethod` | A character string specifying the method for variable selection. Options are `"forward"` for forward selection or `"all"` for using all variables. Default is `"forward"`. |
| `testStat` | A character string specifying the test statistic to use for forward selection. Options are `"Pillai"` or `"Wilks"`. Default is `"Pillai"`. |
| `correction` | A logical value indicating whether to apply a multiple comparison correction during forward selection. Default is `TRUE`. |

| | |
|---|---|
| `alpha` | A numeric value between 0 and 1 specifying the significance level for the test statistic during forward selection. Default is 0.1. |
| `prior` | A numeric vector representing the prior probabilities for each class in the response variable. If `NULL`, the observed class frequencies are used as the prior. Default is `NULL`. |
| `misClassCost` | A square matrix $C$, where each element $C_{ij}$ represents the cost of classifying an observation into class $i$ given that it truly belongs to class $j$. If `NULL`, a default matrix with equal misclassification costs for all class pairs is used. Default is `NULL`. |
| `missingMethod` | A character vector of length 2 specifying how to handle missing values for numerical and categorical variables, respectively. Default is `c("medianFlag", "newLevel")`. |
| `downSampling` | A logical value indicating whether to perform downsampling to balance the class distribution in the training data or speed up the program. Default is `FALSE`. |
| `kSample` | An integer specifying the maximum number of samples to take from each class during downsampling. If `NULL`, the number of samples is limited to the size of the smallest class. Default is `NULL`. |

## Value

A list of class `ULDA` containing the following components:

| | |
|---|---|
| `scaling` | The matrix of scaling coefficients for the linear discriminants. |
| `groupMeans` | The group means of the linear discriminant scores. |
| `prior` | The prior probabilities for each class. |
| `misClassCost` | The misclassification cost matrix. |
| `misReference` | A reference for handling missing values. |
| `terms` | The terms used in the model formula. |
| `xlevels` | The levels of the factors used in the model. |
| `varIdx` | The indices of the selected variables. |
| `varSD` | The standard deviations of the selected variables. |
| `varCenter` | The means of the selected variables. |
| `statPillai` | The Pillai's trace statistic. |
| `pValue` | The p-value associated with Pillai's trace. |
| `predGini` | The Gini index of the predictions on the training data. |
| `confusionMatrix` | The confusion matrix for the training data predictions. |
| `forwardInfo` | Information about the forward selection process, if applicable. |
| `stopInfo` | A message indicating why forward selection stopped, if applicable. |

## References

Howland, P., Jeon, M., & Park, H. (2003). *Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition*. SIAM Journal on Matrix Analysis and Applications

Wang, S. (2024). A New Forward Discriminant Analysis Framework Based On Pillai's Trace and ULDA. *arXiv preprint arXiv:2409.03136*. Available at https://arxiv.org/abs/2409.03136.

## Examples

```
# Fit the ULDA model
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "all")

# Fit the ULDA model with forward selection
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "forward")
```

---

## `forwardSel` *Forward Selection via Multivariate Test Statistics*

---

## Description

This function performs forward selection on a dataset based on multivariate test statistics (`Pillai` or `Wilks`). It iteratively adds variables that contribute the most to the test statistic until no significant variables are found or a stopping criterion is met.

## Usage

```
forwardSel(m, response, testStat = "Pillai", alpha = 0.1, correction = TRUE)
```

## Arguments

| | |
|---|---|
| `m` | A numeric matrix containing the predictor variables. Rows represent observations and columns represent variables. |
| `response` | A factor representing the response variable with multiple levels (groups). |
| `testStat` | A character string specifying the test statistic to use. Can be `"Pillai"` (default) or `"Wilks"`. |
| `alpha` | A numeric value between 0 and 1 specifying the significance level for the test. Default is 0.1. |
| `correction` | A logical value indicating whether to apply a multiple comparison correction. Default is `TRUE`. |

## Value

A list with three components:

| | |
|---|---|
| `currentVarList` | A vector of selected variable indices based on the forward selection process. |
| `forwardInfo` | A data frame containing detailed information about the forward selection process, including the selected variables, test statistics, and thresholds. |
| `stopInfo` | A character string describing why the selection process stopped. |

## References

Wang, S. (2024). A New Forward Discriminant Analysis Framework Based On Pillai's Trace and ULDA. *arXiv preprint arXiv:2409.03136*. Available at `https://arxiv.org/abs/2409.03136`.

---

## getBestVar — *Select Best Variable at Current Step Based on Multivariate Test Statistics*

---

### Description

This function selects the best variable based on the specified multivariate test statistic (`Pillai` or `Wilks`). It evaluates the statistic for each candidate variable in `newVar` when combined with `currentVar`, and returns the index and test statistic of the best variable. It also identifies collinear variables.

### Usage

```
getBestVar(currentVar, newVar, Sw, St, testStat = "Pillai")
```

### Arguments

| | |
|---|---|
| `currentVar` | A numeric vector indicating the indices of currently selected variables. |
| `newVar` | A numeric vector indicating the indices of candidate variables to be tested. |
| `Sw` | A matrix representing the within-class scatter matrix. |
| `St` | A matrix representing the total scatter matrix. |
| `testStat` | A character string specifying the test statistic to use. Can be either `"Pillai"` or `"Wilks"`. Default is `"Pillai"`. |

### Value

A list containing:

| | |
|---|---|
| `stopflag` | A logical value indicating whether the best variable is collinear (i.e., should the selection stop). |
| `varIdx` | The index of the selected variable from `newVar` based on the test statistic. |
| `stat` | The value of the test statistic for the selected variable. |
| `collinearVar` | A vector of indices from `newVar` representing collinear variables. |

## `getDataInShape` *Align Data with a Missing Reference*

### Description

This function aligns a given dataset (`data`) with a reference dataset (`missingReference`). It ensures that the structure, column names, and factor levels in `data` match the structure of `missingReference`. If necessary, missing columns are initialized with `NA`, and factor levels are adjusted to match the reference. Additionally, it handles the imputation of missing values based on the reference and manages flag variables for categorical or numerical columns.

### Usage

```
getDataInShape(data, missingReference)
```

### Arguments

| | |
|---|---|
| `data` | A data frame to be aligned and adjusted according to the `missingReference`. |
| `missingReference` | A reference data frame that provides the structure (column names, factor levels, and missing value reference) for aligning `data`. |

### Value

A data frame where the structure, column names, and factor levels of `data` are aligned with `missingReference`. Missing values in `data` are imputed based on the first row of the `missingReference`, and flag variables are updated accordingly.

### Examples

```
data <- data.frame(
  X1_FLAG = c(0, 0, 0),
  X1 = factor(c(NA, "C", "B"), levels = LETTERS[2:3]),
  X2_FLAG = c(NA, 0, 1),
  X2 = c(2, NA, 3)
)

missingReference <- data.frame(
  X1_FLAG = 1,
  X1 = factor("A", levels = LETTERS[1:2]),
  X2 = 1,
  X2_FLAG = 1
)

getDataInShape(data, missingReference)
```

## `getDesignMatrix` *Generate the Design Matrix for LDA Model*

### Description

Generate the Design Matrix for LDA Model

### Usage

```
getDesignMatrix(modelLDA, data, scale = FALSE)
```

### Arguments

| | |
|---|---|
| `modelLDA` | A fitted LDA model object containing the terms, variable indices, variable centers, and scaling factors. |
| `data` | A data frame containing the predictor variables that are used to create the design matrix. |
| `scale` | A logical value indicating whether to scale the design matrix based on the mean and standard deviation of the variables (default is `FALSE`). |

### Value

A design matrix where each row corresponds to an observation and each column to a predictor variable. If `scale = TRUE`, the variables are centered and scaled based on the means and standard deviations provided in the LDA model object.

## `getDownSampleInd` *Helper Function to Generate Training Set Indices Through Downsampling*

### Description

This function selects the indices for the training set based on the class vector `response`. It allows for optional downsampling to balance the class distribution by limiting the number of samples per class.

### Usage

```
getDownSampleInd(response, downSampling = FALSE, kSample = NULL)
```

### Arguments

| | |
|---|---|
| `response` | A factor vector representing the class labels. |
| `downSampling` | A logical value indicating whether downsampling should be applied. If `TRUE`, downsampling is performed to limit the number of samples per class based on `kSample`. Note that this may not result in equal class frequencies, as `kSample` defines an upper limit for each class, not a lower limit. |
| `kSample` | An integer specifying the maximum number of samples to be selected per class. If `NULL`, the number of samples is limited to the size of the smallest class. |

### Value

An integer vector of indices representing the selected samples from the original `response` vector.

---

## `getLDscores` *Compute Linear Discriminant Scores*

---

### Description

Compute Linear Discriminant Scores

### Usage

```
getLDscores(modelLDA, data, nScores = -1)
```

### Arguments

| | |
|---|---|
| `modelLDA` | A fitted LDA model object containing the scaling matrix and the reference structure for missing data. |
| `data` | A data frame containing the predictor variables for which to compute the linear discriminant scores. |
| `nScores` | An integer specifying the number of discriminant scores to compute. If `-1` (default), all scores are computed. |

### Value

A matrix of linear discriminant scores, where rows correspond to observations and columns correspond to the computed discriminant scores. If `nScores > 0`, only the specified number of scores is returned; otherwise, all scores are computed and returned.

## `getMode` *Calculate the Mode of a Factor Variable with Optional Priors*

### Description

Calculate the Mode of a Factor Variable with Optional Priors

### Usage

```
getMode(v, prior)
```

### Arguments

| | |
|---|---|
| `v` | A factor or vector that can be coerced into a factor. The mode will be calculated from the levels of this factor. |
| `prior` | A numeric vector of prior weights for each level of the factor. |

### Value

The mode of the factor `v` as a character string. If all values are `NA`, the function returns `NA`.

## `getNumFlag` *Identify Numeric, Integer, or Logical Columns in a Data Frame*

### Description

This function checks whether the columns in a data frame (or a vector) are of type numeric, integer, or logical. It can return a logical vector indicating whether each column matches these types, or, if `index = TRUE`, it returns the indices of the matching columns.

### Usage

```
getNumFlag(data, index = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | A data frame or a vector. The function will check the data types of the columns (if `data` is a data frame) or the type of the vector. |
| `index` | A logical value. If `FALSE` (default), the function returns a logical vector indicating which columns are numeric, integer, or logical. If `TRUE`, it returns the indices of these columns. |

### Value

If `index = FALSE` (default), the function returns a logical vector with one element for each column (or the vector itself), where `TRUE` indicates that the column is of type numeric, integer, or logical, and `FALSE` indicates it is not. If `index = TRUE`, the function returns an integer vector containing the indices of the columns that are numeric, integer, or logical.

---

## `missingFix` *Impute Missing Values and Add Missing Flags to a Data Frame*

---

### Description

This function imputes missing values in a data frame based on specified methods for numerical and categorical variables. Additionally, it can add flag columns to indicate missing values. For numerical variables, missing values can be imputed using the mean or median. For categorical variables, missing values can be imputed using the mode or a new level. This function also removes constant columns (all NAs or all observed but the same value).

### Usage

```
missingFix(data, missingMethod = c("medianFlag", "newLevel"))
```

### Arguments

| | |
|---|---|
| `data` | A data frame containing the data to be processed. Missing values (`NA`) will be imputed based on the methods provided in `missingMethod`. |
| `missingMethod` | A character vector of length 2 specifying the methods for imputing missing values. The first element specifies the method for numerical variables (`"mean"`, `"median"`, `"meanFlag"`, or `"medianFlag"`), and the second element specifies the method for categorical variables (`"mode"`, `"modeFlag"`, or `"newLevel"`). If `"Flag"` is included, a flag column will be added for the corresponding variable type. |

### Value

A list with two elements:

| | |
|---|---|
| `data` | The original data frame with missing values imputed, and flag columns added if applicable. |
| `ref` | A reference row containing the imputed values and flag levels, which can be used for future predictions or reference. |

## Examples

```
dat <- data.frame(
  X1 = rep(NA, 5),
  X2 = factor(rep(NA, 5), levels = LETTERS[1:3]),
  X3 = 1:5,
  X4 = LETTERS[1:5],
  X5 = c(NA, 2, 3, 10, NA),
  X6 = factor(c("A", NA, NA, "B", "B"), levels = LETTERS[1:3])
)
missingFix(dat)
```

---

## `nonConstInd` *Identify Non-Constant Columns in a Data Frame*

---

### Description

Identify Non-Constant Columns in a Data Frame

### Usage

```
nonConstInd(data, tol = 1e-08, na.rm = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | A data frame in which columns will be checked for constant values. Columns can be of any type (numeric, integer, logical, or factor). |
| `tol` | A numeric tolerance value (default is `1e-8`) that applies to numerical columns. |
| `na.rm` | A logical value. If `FALSE` (default), missing values are retained during the check. |

### Value

An integer vector containing the indices of the non-constant columns in the data frame. If all columns are constant, an empty vector is returned.

---

## `plot.ULDA` *Plot Decision Boundaries and Linear Discriminant Scores*

---

### Description

This function plots the decision boundaries and linear discriminant (LD) scores for a given ULDA model. If it is a binary classification problem, a density plot is created. Otherwise, a scatter plot with decision boundaries is generated.

## Usage

```
## S3 method for class 'ULDA'
plot(x, datX, response, ...)
```

## Arguments

| | |
|---|---|
| `x` | A fitted ULDA model object. |
| `datX` | A data frame containing the predictor variables. |
| `response` | A factor representing the response variable (training labels) corresponding to `datX`. |
| `...` | Additional arguments. |

## Value

A `ggplot2` plot object, either a density plot or a scatter plot with decision boundaries.

## Examples

```
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "all")
plot(fit, iris[, -5], iris[, 5])
```

---

`predict.ULDA` *Predict Method for ULDA Model*

---

## Description

This function predicts the class labels or class probabilities for new data using a fitted ULDA model. The prediction can return either the most likely class (`"response"`) or the posterior probabilities for each class (`"prob"`).

## Usage

```
## S3 method for class 'ULDA'
predict(object, newdata, type = c("response", "prob"), ...)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `ULDA` model object. |
| `newdata` | A data frame containing the new predictor variables for which predictions are to be made. |
| `type` | A character string specifying the type of prediction to return. `"response"` returns the predicted class labels, while `"prob"` returns the posterior probabilities for each class. Default is `"response"`. |
| `...` | Additional arguments. |

## Value

If `type = "response"`, the function returns a vector of predicted class labels. If `type = "prob"`, it returns a matrix of posterior probabilities, where each row corresponds to a sample and each column to a class.

## Examples

```
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "all")

# Predict class labels
predictions <- predict(fit, iris, type = "response")

# Predict class probabilities
prob_predictions <- predict(fit, iris, type = "prob")
```

# Index