Package ‘manydist’

February 12, 2025

Type Package

Title Unbiased Distances for Mixed-Type Data

Version 0.4.3

Maintainer Angelos Markos <amarkos@gmail.com>

Description A comprehensive framework for calculating unbiased distances in datasets containing mixed-type variables (numerical and categorical). The package implements a general formulation that ensures multivariate additivity and commensurability, meaning that variables contribute equally to the overall distance regardless of their type, scale, or distribution. Supports multiple distance measures including Gower's distance, Euclidean distance, Manhattan distance, and various categorical variable distances such as simple matching, Eskin, occurrence frequency, and association-based distances. Provides tools for variable scaling (standard deviation, range, robust range, and principal component scaling), and handles both independent and association-based category dissimilarities. Implements methods to correct for biases that typically arise from different variable types, distributions, and number of categories. Particularly useful for cluster analysis, data visualization, and other distance-based methods when working with mixed data. Methods based on van de Velden et al. (2024) <doi:10.48550/arXiv.2411.00429> ``Unbiased mixed variables distance''.

Imports entropy, Matrix, fastDummies, data.table, philentropy, cluster, purrr, dplyr, tidyr, forcats, tibble, magrittr, fpc, recipes, rsample, Rfast, readr, distances

Depends R (>= 4.1.0)

Suggests palmerpenguins

License GPL-3

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Alfonso Iodice D'Enza [aut],
Angelos Markos [aut, cre],
Michel van de Velden [aut],
Carlo Cavicchia [aut]

Repository CRAN

Date/Publication 2025-02-12 19:40:02 UTC

Contents

`cdist` *Calculation of Pairwise Distances for Categorical Data*

Description

Computes a distance matrix for categorical variables with support for validation data, multiple distance metrics, and variable weighting. The function implements various distance calculation approaches as described in van de Velden et al. (2024), including commensurable distances and supervised options when response variable is provided.

Usage

```
cdist(x, response = NULL, validate_x = NULL, method = "tot_var_dist",
      commensurable = FALSE, weights = 1)
```

Arguments

`x`	A data frame or matrix of categorical variables (factors).
`response`	Optional response variable for supervised distance calculations. Default is `NULL`.
`validate_x`	Optional validation data frame or matrix. If provided, distances are computed between observations in `validate_x` and `x`. Default is `NULL`.
`method`	Character string or vector specifying the distance metric(s). Options include:

- `"tot_var_dist"`: Total variation distance (default)
- `"HL"`, `"HLeucl"`: Hennig-Liao distance
- `"cat_dis"`: Category-based dissimilarity
- `"mca"`: Multiple correspondence analysis based
- `"st_dev"`: Standard deviation based
- `"matching"`, `"eskin"`, `"iof"`, `"of"`: Various coefficients
- `"goodall_3"`, `"goodall_4"`: Goodall-based distances
- `"gifi_chi2"`: Gifi chi-square distance
- `"lin"`: Lin’s similarity measure
- `"var_entropy"`, `"var_mutability"`: Variability-based measures

	• `"supervised"`, `"supervised_full"`: Response-guided distances • `"le_and_ho"`: Le and Ho distance • Additional methods from philentropy package Can be a single string or vector for different methods per variable.
`commensurable`	Logical. If `TRUE`, standardizes each variable's distance matrix by dividing by its mean distance. Default is `FALSE`.
`weights`	Numeric vector or matrix of weights. If vector, must have length equal to number of variables. If matrix, must match the dimension of level-wise distances. Default is 1 (equal weighting).

Details

The `cdist` function provides a comprehensive framework for categorical distance calculations:

- Supports multiple distance calculation methods that can be specified globally or per variable
- Handles validation data through `validate_x` parameter
- Implements supervised distances when response variable is provided
- Supports commensurable distances for better comparability across variables
- Provides flexible weighting schemes at variable and level granularity

Important notes:

- Input variables are automatically converted to factors with dropped unused levels
- Different methods per variable is not supported for `"none"`, `"st_dev"`, `"HL"`, `"cat_dis"`, `"HLeucl"`, `"mca"`
- Weight vector length must match the number of variables when specified as a vector
- Variables should be factors; numeric variables will cause errors

Value

A list containing:

`distance_mat`	Matrix of pairwise distances. If `validate_x` is provided, rows correspond to validation observations and columns to training observations.
`delta`	Matrix or list of matrices containing level-wise distances for each variable.
`delta_names`	Vector of level names used in the delta matrices.

References

van de Velden, M., Iodice D'Enza, A., Markos, A., Cavicchia, C. (2024). (Un)biased distances for mixed-type data. *arXiv preprint*. Retrieved from `https://arxiv.org/abs/2411.00429`.

See Also

`mdist` for mixed-type data distances, `ndist` for continuous data distances

Examples

```
library(palmerpenguins)
library(rsample)

# Prepare data with complete cases for both categorical variables and response
complete_vars <- c("species", "island", "sex", "body_mass_g")
penguins_complete <- penguins[complete.cases(penguins[, complete_vars]), ]
penguins_cat <- penguins_complete[, c("species", "island", "sex")]
response <- penguins_complete$body_mass_g

# Create training-test split
set.seed(123)
penguins_split <- initial_split(penguins_cat, prop = 0.8)
tr_penguins <- training(penguins_split)
ts_penguins <- testing(penguins_split)
response_tr <- response[penguins_split$in_id]
response_ts <- response[-penguins_split$in_id]

# Basic usage
result <- cdist(tr_penguins)

# With validation data
val_result <- cdist(x = tr_penguins,
                   validate_x = ts_penguins,
                   method = "tot_var_dist")

# ...and commensurability
val_result_COMM <- cdist(x = tr_penguins,
                   validate_x = ts_penguins,
                   method = "tot_var_dist",
                   commensurable = TRUE)

# Supervised distance with response variable
sup_result <- cdist(x = tr_penguins,
                   response = response_tr,
                   method = "supervised")

# Supervised with validation data
sup_val_result <- cdist(x = tr_penguins,
                       validate_x = ts_penguins,
                       response = response_tr,
                       method = "supervised")

# Commensurable distances with custom weights
comm_result <- cdist(tr_penguins,
                    commensurable = TRUE,
                    weights = c(2, 1, 1))

# Different methods per variable
multi_method <- cdist(tr_penguins,
                     method = c("matching", "goodall_3", "tot_var_dist"))
```

`fifa_nl` *FIFA 21 Player Data - Dutch League*

Description

The `fifa_nl` dataset contains information on players in the Dutch League from the FIFA 21 video game. This dataset includes various attributes of players, such as demographics, club details, skill ratings, and physical characteristics.

Usage

```
data("fifa_nl")
```

Format

A data frame with observations on various attributes describing the players.

`player_positions` Primary playing positions of the player.

`nationality` The country the player represents.

`team_position` Player's assigned position within their club.

`club_name` Name of the club the player is part of.

`work_rate` The player's work rate, describing defensive and attacking intensity.

`weak_foot` Skill rating for the player's non-dominant foot, ranging from 1 to 5.

`skill_moves` Skill moves rating, indicating technical skill and ability to perform complex moves, on a scale of 1 to 5.

`international_reputation` Player's reputation on an international scale, from 1=local to 3=global star.

`body_type` Body type of the player (Lean, Normal, Stocky.

`preferred_foot` Dominant foot of the player, either Left or Right.

`age` Age of the player in years.

`height_cm` Height of the player in centimeters.

`weight_kg` Weight of the player in kilograms.

`overall` Overall skill rating of the player out of 100.

`potential` Potential skill rating the player may achieve in the future.

`value_eur` Estimated market value of the player in Euros.

`wage_eur` Player's weekly wage in Euros.

`release_clause_eur` Release clause value in Euros, which other clubs must pay to buy out the player's contract.

`pace` Speed rating of the player out of 100.

`shooting` Shooting skill rating out of 100.

`passing` Passing skill rating out of 100.

`dribbling` Dribbling skill rating out of 100.

`defending` Defending skill rating out of 100.

`physic` Physicality rating out of 100.

Details

This dataset provides a snapshot of player attributes and performance indicators as represented in FIFA 21 for players in the Dutch League. It can be used to analyze player characteristics, compare skills across players, and explore potential relationships among variables such as age, position, and various skill ratings.

References

Stefano Leone. (2021). *FIFA 21 Complete Player Dataset*. Retrieved from `https://www.kaggle.com/datasets/stefanoleone992/fifa-21-complete-player-dataset`.

Examples

```
data(fifa_nl)
summary(fifa_nl)
```

`mdist` *Calculation of Pairwise Distances for Mixed-Type Data*

Description

Computes pairwise distances between observations described by numeric and/or categorical attributes, with support for validation data. The function provides options for computing independent, dependent, and practice-based distances as defined in van de Velden et al. (2024), with support for various continuous and categorical distance metrics, scaling, and commensurability adjustments.

Usage

```
mdist(x, validate_x = NULL, response = NULL, distance_cont = "manhattan",
      distance_cat = "tot_var_dist", commensurable = FALSE, scaling = "none",
      ncomp = ncol(x), threshold = NULL, preset = "custom")
```

Arguments

`x`	A dataframe or tibble containing continuous (coded as numeric), categorical (coded as factors), or mixed-type variables.
`validate_x`	Optional validation data with the same structure as `x`. If provided, distances are computed between observations in `validate_x` and `x`. Default is `NULL`.
`response`	An optional factor for supervised distance calculation in categorical variables, applied only if `distance_cat = "supervised"`. Default is `NULL`.
`distance_cont`	Character string specifying the distance metric for continuous variables. Options include `"manhattan"` (default) and `"euclidean"`.
`distance_cat`	Character string specifying the distance metric for categorical variables. Options include `"tot_var_dist"` (default), `"HL"`, `"HLeucl"`, `cat_dis`, `mca`, `st_dev`, `"matching"`, `"eskin"`, `"iof"`, `"of"`, `"goodall_3"`, `"goodall_4"`, `"gifi_chi2"`, `"lin"`, `"var_entropy"`, `"var_mutability"`, `"supervised"`, `"supervised_full"`, `"le_and_ho"` and all the options in the package philentropy.

`commensurable`	Logical. If `TRUE`, the function adjusts each variable's contribution to ensure equal average influence in the overall distance. Default is `FALSE`.
`scaling`	Character string specifying the scaling method for continuous variables. Options include `"none"` (default), `"std"`, `"range"`, `"pc_scores"`, and `"robust"`.
`ncomp`	Integer specifying the number of components to retain when `scaling = "pc_scores"`. Default is `ncol(x)`.
`threshold`	Numeric value specifying the percentage of variance explained by retained components when `scaling = "pc_scores"`. Overrides `ncomp` if specified. Default is `NULL`.
`preset`	Character string specifying pre-defined combinations of arguments. Options include: • `"custom"` (default): Use specified distance metrics and parameters • `"gower"`: Gower's distance for mixed data • `"unbiased_dependent"`: Total variation distance for categorical and Manhattan for standardized continuous • `"euclidean_onehot"`: Euclidean distance on one-hot encoded categorical and standardized continuous • `"catdissim"`: Matching distance for categorical and Manhattan for standardized continuous

Value

A matrix of pairwise distances. If `validate_x` is provided, rows correspond to validation observations and columns to training observations.

References

van de Velden, M., Iodice D'Enza, A., Markos, A., Cavicchia, C. (2024). (Un)biased distances for mixed-type data. *arXiv preprint*. Retrieved from `https://arxiv.org/abs/2411.00429`.

See Also

`cdist` for categorical-only distances, `ndist` for continuous-only distances

Examples

```
library(palmerpenguins)
library(rsample)

# Prepare complete data
pengmix <- palmerpenguins::penguins[complete.cases(palmerpenguins::penguins), ]

# Create training-test split
set.seed(123)
pengmix_split <- initial_split(pengmix, prop = 0.8)
tr_pengmix <- training(pengmix_split)
ts_pengmix <- testing(pengmix_split)
```

```
# Example 1: Basic usage with validation data
dist_matrix <- mdist(x = tr_pengmix,
                     validate_x = ts_pengmix)

# Example 2: Gower preset with validation
dist_gower <- mdist(x = tr_pengmix,
                    validate_x = ts_pengmix,
                    preset = "gower",
                    commensurable = TRUE)

# Example 3: Euclidean one-hot preset with validation
dist_onehot <- mdist(x = tr_pengmix,
                     validate_x = ts_pengmix,
                     preset = "euclidean_onehot")

# Example 4: Custom preset with standardization
dist_custom <- mdist(x = tr_pengmix,
                     validate_x = ts_pengmix,
                     preset = "custom",
                     distance_cont = "manhattan",
                     distance_cat = "matching",
                     commensurable = TRUE,
                     scaling = "std")

# Example 5: PCA-based scaling with threshold
dist_pca <- mdist(x = tr_pengmix,
                  validate_x = ts_pengmix,
                  distance_cont = "euclidean",
                  scaling = "pc_scores",
                  threshold = 0.85)

# Example 6: Categorical variables only
cat_vars <- c("species", "island", "sex")
dist_cat <- mdist(tr_pengmix[, cat_vars],
                  validate_x = ts_pengmix[, cat_vars],
                  distance_cat = "tot_var_dist")

# Example 7: Continuous variables only
num_vars <- c("bill_length_mm", "bill_depth_mm",
              "flipper_length_mm", "body_mass_g")
dist_cont <- mdist(tr_pengmix[, num_vars],
                   validate_x = ts_pengmix[, num_vars],
                   distance_cont = "manhattan",
                   scaling = "std")

# Example 8: Supervised distance with response
response_tr <- tr_pengmix$body_mass_g
dist_sup <- mdist(tr_pengmix,
                  validate_x = ts_pengmix,
                  response = response_tr,
                  distance_cat = "supervised")
```

`ndist` *Calculation of Pairwise Distances for Continuous Data*

Description

Computes a distance matrix for continuous data with support for multiple distance metrics, scaling methods, dimensionality reduction, and validation data. The function implements various distance calculation approaches as described in van de Velden et al. (2024), including options for commensurable distances and variable weighting.

Usage

```
ndist(x, validate_x = NULL, commensurable = FALSE, method = "manhattan",
      sig = NULL, scaling = "none", ncomp = ncol(x), threshold = NULL,
      weights = rep(1, ncol(x)))
```

Arguments

`x`	A data frame or matrix of continuous input variables.
`validate_x`	Optional data frame or matrix for validation data. If provided, distances are computed between observations in `validate_x` and `x`. Default is `NULL`.
`commensurable`	Logical. If `TRUE`, standardizes each variable's distance matrix by dividing by its mean distance, making distances comparable across variables. Default is `FALSE`.
`method`	Character string specifying the distance metric. Options include `"manhattan"`, `"euclidean"`, and `"mahalanobis"`. Default is `"manhattan"`.
`sig`	Covariance matrix to be used when `method = "mahalanobis"`. If `NULL`, computed from the data. Default is `NULL`.
`scaling`	Character string specifying the scaling method. Options: • `"none"`: No scaling • `"std"`: Standardization (zero mean, unit variance) • `"range"`: Min-max scaling to [0,1] • `"pc_scores"`: PCA-based dimensionality reduction • `"robust"`: Robust scaling using median and IQR. Default is `"none"`.
`ncomp`	Number of principal components to retain when `scaling = "pc_scores"`. Default is the number of columns in `x`.
`threshold`	Proportion of variance to retain when `scaling = "pc_scores"`. If specified, overrides `ncomp`. Default is `NULL`.
`weights`	Numeric vector of weights for each variable. Must have length equal to the number of variables in `x`. Default is a vector of ones.

Details

The `ndist` function provides a comprehensive framework for distance calculations in continuous data:

- When `validate_x` is provided, computes distances between observations in `validate_x` and `x`.
- Supports multiple scaling methods that can be applied before distance calculation.
- PCA-based dimensionality reduction can be controlled either by number of components or variance threshold.
- For Mahalanobis distance, handles singular covariance matrices with appropriate error messages.
- Implements commensurable distances for better comparability across variables.

Warning: The function validates:

- Weight vector length must match the number of variables
- Covariance matrix singularity for Mahalanobis distance
- Compatibility of `x` and `validate_x` dimensions

Value

A distance matrix where element [i,j] represents the distance between:

- observation i and j of `x` if `validate_x` is `NULL`
- observation i of `validate_x` and observation j of `x` if `validate_x` is provided

References

van de Velden, M., Iodice D'Enza, A., Markos, A., Cavicchia, C. (2024). (Un)biased distances for mixed-type data. *arXiv preprint*. Retrieved from `https://arxiv.org/abs/2411.00429`.

See Also

`mdist` for mixed-type data distances, `cdist` for categorical data distances.

Examples

```
library(palmerpenguins)
library(rsample)

penguins_cont <- palmerpenguins::penguins[, c("bill_length_mm",
"bill_depth_mm", "flipper_length_mm", "body_mass_g")]
penguins_cont <- penguins_cont[complete.cases(penguins_cont), ]

# Basic usage
dist_matrix <- ndist(penguins_cont)

# Commensurable distances with standardization
dist_matrix <- ndist(penguins_cont,
                    commensurable = TRUE,
                    scaling = "std")

# PCA-based dimensionality reduction
dist_matrix <- ndist(penguins_cont,
                    scaling = "pc_scores",
                    threshold = 0.95)

# Mahalanobis distance
dist_matrix <- ndist(penguins_cont,
                    method = "mahalanobis")

# Weighted Euclidean distance
dist_matrix <- ndist(penguins_cont,
                    method = "euclidean",
                    weights = c(1, 0.5, 2, 1))

# Training-test split example with validation data
set.seed(123)
# Create training-test split using rsample
penguins_split <- initial_split(penguins_cont, prop = 0.8)
tr_penguins <- training(penguins_split)
ts_penguins <- testing(penguins_split)

# Basic usage with training data only
dist_matrix <- ndist(tr_penguins)

# Computing distances between test and training sets
val_dist_matrix <- ndist(x = tr_penguins,
                        validate_x = ts_penguins,
                        method = "euclidean")

# Using validation data with standardization
val_dist_matrix_std <- ndist(x = tr_penguins,
                            validate_x = ts_penguins,
                            scaling = "std",
                            method = "manhattan")

# Validation with PCA and commensurability
val_dist_matrix_pca <- ndist(x = tr_penguins,
                            validate_x = ts_penguins,
                            scaling = "pc_scores",
                            ncomp = 2,
                            commensurable = TRUE)

# Validation with robust scaling and custom weights
val_dist_matrix_robust <- ndist(x = tr_penguins,
                               validate_x = ts_penguins,
                               scaling = "robust",
                               weights = c(1, 0.5, 2, 1))

# Mahalanobis distance with validation data
val_dist_matrix_mahal <- ndist(x = tr_penguins,
                              validate_x = ts_penguins,
                              method = "mahalanobis")
```

Index