Package 'dbrobust'

May 8, 2026

Type Package

Title Robust Distance-Based Visualization and Analysis of Mixed-Type Data

Version 1.0.0

Date 2025-09-16

Author Marcos Álvarez [aut],
Eva Boj [aut, cre],
Aurea Grané [aut]

Maintainer Eva Boj <evaboj@ub.edu>

Description Robust distance-based methods applied to matrices and data frames, producing distance matrices that can be used as input for various visualization techniques such as graphs, heatmaps, or multidimensional scaling configurations. See Boj and Grané (2024) <doi:10.1016/j.seps.2024.101992>.

License GPL-3

Repository CRAN

Encoding UTF-8

Imports MASS, proxy, ade4, Rdpack, dbstats, StatMatch, RColorBrewer, pheatmap, qgraph, GGally, ggplot2, vegan

RdMacros Rdpack

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.5)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Date/Publication 2025-09-22 07:50:12 UTC

Contents

`calculate_distances` *Compute Distance or Similarity Matrices*

Description

Computes a distance or similarity matrix between rows of a data frame or matrix, supporting a wide variety of distance metrics.

Usage

```
calculate_distances(
  x,
  method = "gower",
  output_format = "dist",
  squared = FALSE,
  p = NULL,
  similarity_transform = "linear",
  ...
)
```

Arguments

`x`	A matrix or data.frame. Each row represents an observation.
`method`	A string specifying the distance/similarity method. Supported: • **Binary**: `"jaccard"`, `"dice"`, `"sokal_michener"`, `"russell_rao"`, `"sokal_sneath"`, `"kulczynski"`,`"hamming"`. • **Categorical**: `"matching_coefficient"`. • **Continuous**: `"euclidean"`, `"euclidean_standardized"`, `"manhattan"`, `"minkowski"`, `"canberra"`, `"maximum"`, `"cosine"`, `"correlation"`, `"mahalanobis"`. • **Mixed**: `"gower"`.
`output_format`	Output format: `"dist"` (distance object), `"matrix"` (numeric matrix), or `"similarity"` (only for binary/categorical/mixed methods).
`squared`	Logical; if `TRUE`, returns squared distances (not applied to similarities).

p — Numeric; the power parameter for the Minkowski distance (required if `method = "minkowski"`).

similarity_transform — Character string; if `output_format = "similarity"`, this specifies the formula to convert distances to similarity scores. Supported:

- `"linear"` (default): $s_{ij} = 1 - \delta_{ij}$
- `"sqrt"`: $s_{ij} = 1 - \delta_{ij}^2$

... — Additional arguments passed to underlying functions.

Details

When `output_format = "similarity"`, the function transforms computed distances into similarity scores using one of the supported transformations.

The similarity transformation options are:

`"linear"` Direct inversion of distance: $s_{ij} = 1 - \delta_{ij}$.

`"sqrt"` Squared distance inversion: $s_{ij} = 1 - \delta_{ij}^2$, which may better preserve Euclidean properties.

Value

Depending on `output_format`, returns:

- dist object (if `output_format = "dist"`)
- numeric matrix (if `output_format = "matrix"` or `"output_format = similarity"`)

See Also

`dist` for basic distance measures, `dist.binary` for binary distances, `dist` for advanced metrics like cosine or correlation

Examples

```
# Load example dataset
data("Data_HC_contamination", package = "dbrobust")
df <- Data_HC_contamination

# --- Quick Example ---
numeric_data <- df[1:10, 1:4]  # subset for speed
d_euclid <- calculate_distances(
  numeric_data,
  method = "euclidean",
  output_format = "matrix"
)

# Load example dataset
data("Data_HC_contamination", package = "dbrobust")
df <- Data_HC_contamination[1:20,]

# Example 1: Euclidean distance (numeric variables only)
numeric_data <- df[, 1:4]
```

```
d_euclid <- calculate_distances(
  numeric_data,
  method = "euclidean",
  output_format = "matrix"
)

# Example 2: Manhattan distance
d_manhattan <- calculate_distances(
  numeric_data,
  method = "manhattan",
  output_format = "matrix"
)

# Example 3: Categorical distance using Matching Coefficient
categorical_data <- df[, 5:7]
d_match <- calculate_distances(
  categorical_data,
  method = "matching_coefficient",
  output_format = "matrix"
)

# Example 4: Mixed data distance using Gower (automatic type detection, asymmetric binary)
d_gower_asym <- calculate_distances(
  df,
  method = "gower",
  output_format = "dist",
  binary_asym = TRUE
)

# Example 5: Minkowski distance with p = 3
d_minkowski <- calculate_distances(
  numeric_data,
  method = "minkowski",
  p = 3,
  output_format = "matrix"
)

# Example 6: Jaccard distance for binary variables
binary_data <- df[, 8:9]
d_jaccard <- calculate_distances(
  binary_data,
  method = "jaccard",
  output_format = "matrix"
)

# Example 7: Mahalanobis distance
d_mahal <- calculate_distances(
  numeric_data,
  method = "mahalanobis",
  output_format = "matrix"
)

# Example 8: Manual selection of variables for Gower distance
```

```
continuous_vars <- 1:4
binary_vars <- 8:9
categorical_vars <- 5:7
d_gower_manual <- calculate_distances(
  df,
  method = "gower",
  output_format = "dist",
  continuous_cols = continuous_vars,
  binary_cols = binary_vars,
  categorical_cols = categorical_vars
)
```

`Data_HC_contamination` *High-correlation dataset with contamination*

Description

Synthetic dataset generated from a multivariate normal distribution with strong correlation structure ($\rho = 0.8$). It contains 550 observations and 10 variables of mixed type (continuous, categorical, binary, and weights). The last 50 rows correspond to contaminated observations created by adding perturbations equal to three times the standard deviation of each quantitative variable to a subset of original units. This results in a controlled 10% contamination level. These data follow the design in (Boj and Grané 2024).

Usage

```
Data_HC_contamination
```

Format

A data frame with 550 rows and 10 variables:

V1 Continuous variable 1

V2 Continuous variable 2

V3 Continuous variable 3

V4 Continuous variable 4

V5 Categorical variable 1 (3 categories, approx. balanced)

V6 Categorical variable 2 (3 categories, approx. balanced)

V7 Categorical variable 3 (4 categories, uniform distribution)

V8 Binary variable 1 (40% zeros, 60% ones)

V9 Binary variable 2 (60% zeros, 40% ones)

w_loop Observation weights derived from the joint distribution of V5 and V8, following a proportional frequency-based scheme.

Details

- Continuous variables were drawn directly from the multivariate normal sample.
- Binary and categorical variables were obtained by discretizing normal margins using percentile-based thresholds.
- Contaminated observations (rows 501–550) were generated by perturbing original cases with fluctuations of 3 SD.
- The weighting scheme prioritizes frequent category combinations.

References

Boj E, Grané A (2024). "The robustification of distance-based linear models: Some proposals." *Socio-Economic Planning Sciences*, **95**, 101992.

`Data_HC_no_contamination`

High-correlation dataset without contamination

Description

Synthetic dataset generated from a multivariate normal distribution with strong correlation structure ($\rho = 0.8$). It contains 500 observations and 10 variables of mixed type (continuous, categorical, binary, and weights). No contaminated cases were added in this version, so the dataset represents a clean scenario with 0% contamination. These data follow the design in (Boj and Grané 2024).

Usage

```
Data_HC_no_contamination
```

Format

A data frame with 500 rows and 10 variables:

V1 Continuous variable 1

V2 Continuous variable 2

V3 Continuous variable 3

V4 Continuous variable 4

V5 Categorical variable 1 (3 categories, approx. balanced)

V6 Categorical variable 2 (3 categories, approx. balanced)

V7 Categorical variable 3 (4 categories, uniform distribution)

V8 Binary variable 1 (40% zeros, 60% ones)

V9 Binary variable 2 (60% zeros, 40% ones)

w_loop Observation weights derived from the joint distribution of V5 and V8, following a proportional frequency-based scheme.

Details

- Continuous variables were drawn directly from the multivariate normal sample.
- Binary and categorical variables were obtained by discretizing normal margins using percentile-based thresholds.
- Unlike other datasets in this collection, no artificial contamination was introduced here.
- The weighting scheme prioritizes frequent category combinations.

References

Boj E, Grané A (2024). "The robustification of distance-based linear models: Some proposals." *Socio-Economic Planning Sciences*, **95**, 101992.

`Data_MC_contamination` *Moderate-correlation dataset with contamination*

Description

Synthetic dataset generated from a multivariate normal distribution with moderate correlation structure ($\rho = 0.6$). It contains 525 observations and 10 variables of mixed type (continuous, categorical, binary, and weights). The last 25 rows correspond to contaminated observations created by adding perturbations equal to three times the standard deviation of each quantitative variable to a subset of original units. This results in a controlled 5% contamination level. These data follow the design in (Boj and Grané 2024).

Usage

```
Data_MC_contamination
```

Format

A data frame with 525 rows and 10 variables:

V1 Continuous variable 1

V2 Continuous variable 2

V3 Continuous variable 3

V4 Continuous variable 4

V5 Categorical variable 1 (3 categories, approx. balanced)

V6 Categorical variable 2 (3 categories, approx. balanced)

V7 Categorical variable 3 (4 categories, uniform distribution)

V8 Binary variable 1 (40% zeros, 60% ones)

V9 Binary variable 2 (60% zeros, 40% ones)

w_loop Observation weights derived from the joint distribution of V5 and V8, following a proportional frequency-based scheme.

Details

- Continuous variables were drawn directly from the multivariate normal sample.
- Binary and categorical variables were obtained by discretizing normal margins using percentile-based thresholds.
- Contaminated observations (rows 501–525) were generated by perturbing original cases with fluctuations of 3 SD.
- The weighting scheme prioritizes frequent category combinations.

References

Boj E, Grané A (2024). "The robustification of distance-based linear models: Some proposals." *Socio-Economic Planning Sciences*, **95**, 101992.

`Data_MC_no_contamination`

Moderate-correlation dataset without contamination

Description

Synthetic dataset generated from a multivariate normal distribution with moderate correlation structure ($\rho = 0.6$). It contains 500 observations and 10 variables of mixed type (continuous, categorical, binary, and weights). No contaminated cases were added in this version, so the dataset represents a clean scenario with 0% contamination. These data follow the design in (Boj and Grané 2024).

Usage

```
Data_MC_no_contamination
```

Format

A data frame with 500 rows and 10 variables:

V1 Continuous variable 1

V2 Continuous variable 2

V3 Continuous variable 3

V4 Continuous variable 4

V5 Categorical variable 1 (3 categories, approx. balanced)

V6 Categorical variable 2 (3 categories, approx. balanced)

V7 Categorical variable 3 (4 categories, uniform distribution)

V8 Binary variable 1 (40% zeros, 60% ones)

V9 Binary variable 2 (60% zeros, 40% ones)

w_loop Observation weights derived from the joint distribution of V5 and V8, following a proportional frequency-based scheme.

Details

- Continuous variables were drawn directly from the multivariate normal sample.
- Binary and categorical variables were obtained by discretizing normal margins using percentile-based thresholds.
- Unlike other datasets in this collection, no artificial contamination was introduced here.
- The weighting scheme prioritizes frequent category combinations.

References

Boj E, Grané A (2024). "The robustification of distance-based linear models: Some proposals." *Socio-Economic Planning Sciences*, **95**, 101992.

`make_euclidean` *Force a Pairwise Squared Distance Matrix to Euclidean Form*

Description

Given a pairwise squared distance matrix D (where $D[i,j] = d(i,j)^2$), this function ensures that D corresponds to a valid Euclidean squared distance matrix. The correction is based on the weighted Gram matrix $G_w = -\frac{1}{2} J_w D J_w^\top$, where $J_w = I_n - \mathbf{1} w^\top$ is the centering matrix defined by the weight vector w.

Usage

```
make_euclidean(D, w, tol = 1e-10)
```

Arguments

`D`	Numeric square matrix (n x n) of pairwise squared distances. Must be symmetric with zeros on the diagonal.
`w`	Numeric vector of weights (length n). Internally normalized to sum to 1.
`tol`	Numeric tolerance for detecting negative eigenvalues (default: 1e-10).

Details

If the smallest eigenvalue $\lambda_{\min}$ of G_w is below the negative tolerance `-tol`, the function corrects D by adding a constant shift to guarantee positive semi-definiteness of the Gram matrix, following the approach of (Lingoes 1971) and (Mardia 1978):

$$D_{\text{new}} = D + 2c\mathbf{1}\mathbf{1}^\top - 2cI_n,$$

where $c = |\lambda_{\min}|$.

Value

A list with components:

D_euc Corrected pairwise squared Euclidean distance matrix (n x n).

eigvals_before Eigenvalues of the weighted Gram matrix before correction.

eigvals_after Eigenvalues of the weighted Gram matrix after correction.

transformed Logical, TRUE if correction was applied, FALSE otherwise.

References

Lingoes JC (1971). "Some boundary conditions for a monotone analysis of symmetric matrices." *Psychometrika*, **36**(2), 195–203. Mardia KV (1978). "Some properties of clasical multi-dimesional scaling." *Communications in Statistics-Theory and Methods*, **7**(13), 1233–1241.

See Also

dist, eigen, cmdscale

Examples

```
# Load example dataset
data("Data_HC_contamination")

# Reduce dataset to first 50 rows
Data_small <- Data_HC_contamination[1:50, ]

# Select only continuous variables
cont_vars <- names(Data_small)[1:4]
Data_cont <- Data_small[, cont_vars]

# Compute squared Euclidean distance matrix
dist_mat <- as.matrix(dist(Data_cont))^2

# Introduce a small non-Euclidean distortion
dist_mat[1, 2] <- dist_mat[1, 2] * 0.5
dist_mat[2, 1] <- dist_mat[1, 2]

# Uniform weights
weights <- rep(1, nrow(Data_cont))

# Apply Euclidean correction
res <- make_euclidean(dist_mat, weights)

# Check results (minimum eigenvalues before/after)
res$transformed
min(res$eigvals_before)
min(res$eigvals_after)

# First 5x5 block of corrected matrix
round(res$D_euc[1:5, 1:5], 4)
```

`robust_distances` *Compute Robust Squared Distances for Mixed Data*

Description

Computes a weighted, robust squared distance matrix for datasets containing continuous, binary, and categorical variables. Continuous variables are handled via a robust Mahalanobis distance, and binary and categorical variables are transformed via similarity coefficients. The output is suitable for Euclidean correction with `make_euclidean`.

Usage

```
robust_distances(
  data = NULL,
  cont_vars = NULL,
  bin_vars = NULL,
  cat_vars = NULL,
  w = NULL,
  p = NULL,
  method = c("ggower", "relms"),
  robust_cov = NULL,
  alpha = 0.1,
  return_dist = FALSE
)
```

Arguments

`data`	Data frame or numeric matrix containing the observations.
`cont_vars`	Character vector of column names for continuous variables.
`bin_vars`	Character vector of column names for binary variables.
`cat_vars`	Character vector of column names for categorical variables.
`w`	Numeric vector of observation weights. If NULL, uniform weights are used.
`p`	Integer vector of length 3: `c(#cont, #binary, #categorical)`. Overrides variable type selection if provided.
`method`	Character string: either `"ggower"` or `"relms"` for distance computation.
`robust_cov`	Optional. Precomputed robust covariance matrix for continuous variables. If NULL, it will be estimated internally using the specified trimming proportion `alpha`.
`alpha`	Numeric trimming proportion for robust covariance of continuous variables.
`return_dist`	Logical. If TRUE, returns an object of class `dist`; otherwise, returns a squared distance matrix.

Value

A numeric matrix of squared robust distances (n x n) or a `dist` object if `return_dist = TRUE`.

Examples

```
# Example: Robust Squared Distances for Mixed Data

# Load example data and subset
data("Data_HC_contamination", package = "dbrobust")
Data_small <- Data_HC_contamination[1:50, ]

# Define variable types
cont_vars <- c("V1", "V2", "V3", "V4")  # continuous
cat_vars  <- c("V5", "V6", "V7")        # categorical
bin_vars  <- c("V8", "V9")              # binary

# Use column w_loop as weights
w <- Data_small$w_loop

# --------------------------------
# Method 1: Gower distances
# --------------------------------
dist_sq_ggower <- robust_distances(
  data = Data_small,
  cont_vars = cont_vars,
  bin_vars  = bin_vars,
  cat_vars  = cat_vars,
  w = w,
  alpha = 0.10,
  method = "ggower"
)

# Apply Euclidean correction if needed
res_ggower <- make_euclidean(dist_sq_ggower, w)

# Show first 5x5 block of original and corrected distances
cat("GGower original squared distances (5x5 block):\n")
print(round(dist_sq_ggower[1:5, 1:5], 4))
cat("\nGGower corrected squared distances (5x5 block):\n")
print(round(res_ggower$D_euc[1:5, 1:5], 4))

# --------------------------------
# Method 2: RelMS distances
# --------------------------------
dist_sq_relms <- robust_distances(
  data = Data_small,
  cont_vars = cont_vars,
  bin_vars  = bin_vars,
  cat_vars  = cat_vars,
  w = w,
  alpha = 0.10,
  method = "relms"
)

# Apply Euclidean correction if needed
res_relms <- make_euclidean(dist_sq_relms, w)
```

```
# Show first 5x5 block of original and corrected distances
cat("RelMS original squared distances (5x5 block):\n")
print(round(dist_sq_relms[1:5, 1:5], 4))
cat("\nRelMS corrected squared distances (5x5 block):\n")
print(round(res_relms$D_euc[1:5, 1:5], 4))
```

`visualize_distances` *Visualize Distance Matrices via MDS, Heatmap, or Network Graph*

Description

This function provides a unified interface to visualize distance matrices using classical or weighted Multidimensional Scaling (MDS), heatmaps, or network graphs. Group annotations can be provided for coloring.

Usage

```
visualize_distances(
  dist_mat,
  method = c("mds_classic", "mds_weighted", "heatmap", "qgraph"),
  k = 3,
  weights = NULL,
  group = NULL,
  main_title = NULL,
  tol = 1e-10,
  ...
)
```

Arguments

`dist_mat`	A square distance matrix (numeric matrix) or a `dist` object.
`method`	Character string specifying the visualization method. Options are: • `"mds_classic"`: Classical MDS (cmdscale). • `"mds_weighted"`: Weighted MDS (wcmdscale, requires `weights`). • `"heatmap"`: Heatmap with optional clustering and group annotations. • `"qgraph"`: Network graph representation of similarity.
`k`	Integer. Number of dimensions to retain for MDS (default 3). Must be `>=1` and `<= min(4, n_obs-1)`.
`weights`	Optional numeric vector of weights for weighted MDS. Must match the number of observations.
`group`	Optional factor or vector indicating group membership for coloring plots.
`main_title`	Optional character string specifying the main title of the plot.
`tol`	Numeric tolerance for checking approximate symmetry (default 1e-10).
`...`	Additional arguments passed to internal plotting functions (`plot_heatmap` or `plot_qgraph`).

Details

`visualize_distances` is a wrapper around three internal plotting functions:

- `plot_mds`: Creates a pairwise scatterplot matrix of MDS coordinates with density plots on the diagonal.
- `plot_heatmap`: Plots a heatmap of the distance matrix with hierarchical clustering and optional group annotations.
- `plot_qgraph`: Plots a network graph where nodes represent observations and edges represent similarity.

The function validates that `dist_mat` is square, symmetric, and has zero diagonal elements. If a distance matrix has a `trimmed_idx` attribute and `group` is not provided, a factor indicating "Trimmed" vs "Outlier" is created automatically.

Value

The plotting object is returned and automatically printed:

- MDS plots return a `ggmatrix` from `GGally`.
- Heatmaps return a `pheatmap` object.
- Network graphs are plotted directly (returns `NULL`).

See Also

`cmdscale` for classical MDS. `wcmdscale` for weighted MDS. `pheatmap` for heatmaps. `qgraph` for network graphs. `ggpairs` for MDS scatterplot matrices.

Examples

```
# Load iris dataset
data(iris)

# Compute Euclidean distances on numeric columns
dist_iris <- dist(iris[, 1:4])

# Create a grouping factor based on Species
group_species <- iris$Species

# ---------------------------------------
# Classical MDS (2D)
# ---------------------------------------
visualize_distances(
  dist_mat = dist_iris,
  method = "mds_classic",
  k = 2,
  group = group_species,
  main_title = "Classical MDS - Iris Dataset - Euclidean Distance"
)

# ---------------------------------------
```

```
# Weighted MDS (uniform weights)
# ------------------------------------
weights <- rep(1, nrow(iris))
visualize_distances(
  dist_mat = dist_iris,
  method = "mds_weighted",
  k = 2,
  weights = weights,
  group = group_species,
  main_title = "Weighted MDS - Iris Dataset - Euclidean Distance"
)

# ------------------------------------
# Heatmap (limit rows to 30)
# ------------------------------------
visualize_distances(
  dist_mat = dist_iris,
  method = "heatmap",
  group = group_species,
  main_title = "Iris Heatmap by Species - Euclidean Distance",
  max_n = 30,
  palette = "YlGnBu",
  clustering_method = "complete",
  annotation_legend = TRUE,
  stratified_sampling = TRUE,
  seed = 123
)

# ------------------------------------
# Network Graph (limit nodes to 30)
# ------------------------------------
visualize_distances(
  dist_mat = dist_iris,
  method = "qgraph",
  group = group_species,
  max_nodes = 30,
  label_size = 2,
  edge_threshold = 0.1,
  layout = "spring",
  seed = 123,
  main_title = "Iris Network Graph by Species - Euclidean Distance"
)
```

Index