

Package: dbrobust (via r-universe)

September 26, 2026

Type Package

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

Version 1.1.1

Date 2026-07-28

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](https://doi.org/10.1016/j.seps.2024.101992)>, Boj et al. (2026) <[doi:10.1007/s10260-026-00860-1](https://doi.org/10.1007/s10260-026-00860-1)>, and Grané et al. (2026) <[doi:10.1007/s10182-026-00565-6](https://doi.org/10.1007/s10182-026-00565-6)>.

License GPL-3

Encoding UTF-8

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

RdMacros Rdpack

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

LinkingTo Rcpp, RcppArmadillo

Repository <https://cran.r-universe.dev>

Date/Publication 2026-07-28 09:30:10 UTC

RemoteUrl <https://github.com/cran/dbrobust>

RemoteRef HEAD
RemoteSha ebfbd7c6ed1bc0cf0c89b2f5a773ee89d463e42

Contents

calculate_distances	2
Data_HC_contamination	5
Data_HC_no_contamination	7
Data_MC_contamination	8
Data_MC_no_contamination	9
make_euclidean	10
merge_distances	11
robust_distances	14
visualize_distances	17
Index	21

calculate_distances	<i>Compute Distance or Similarity Matrices</i>
---------------------	--

Description

Computes a distance or similarity matrix between rows of a data frame or matrix, supporting a wide variety of distance metrics for binary, categorical, continuous, and mixed-type data.

Usage

```
calculate_distances(  
  x,  
  method = "gower",  
  output_format = "dist",  
  squared = FALSE,  
  q = NULL,  
  similarity_transform = "linear",  
  type = list(),  
  weights = NULL  
)
```

Arguments

- | | |
|--------|--|
| x | A matrix or data.frame. Each row represents an observation. |
| method | A string specifying the distance/similarity method. Supported options are: <ul style="list-style-type: none">• Binary: "jaccard", "dice", "sokal_michener", "russell_rao", "sokal_sneath", "kulczynski", "hamming".• Categorical: "matching_coefficient", "cramer". |

	• 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).
q	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) or "sqrt".
type	Optional list specifying variable types for Gower distance (passed directly to <code>cluster::daisy</code>). Arguments can be <code>symm</code> , <code>asymm</code> , <code>ordratio</code> , or <code>logratio</code> .
weights	Optional numeric vector containing weights. Its interpretation depends on the chosen method: • For Gower distance (method = "gower"): A vector of length <code>ncol(x)</code> to weight the <i>variables</i> (columns). • For Cramér's V distance (method = "cramer"): A vector of length <code>nrow(x)</code> to weight the <i>observations</i> (rows), useful for handling class imbalance or stratified sampling.

Details

Geometrical Note on Gower Distance: Distance matrices computed via Gower's coefficient (method = "gower") may not satisfy the Euclidean property. Users should exercise caution when passing these distances to downstream algorithms that strictly assume Euclidean geometry.

Handling of Sparsity in Binary Metrics: For binary methods like Jaccard or Dice, pairs of empty vectors (all-zero rows) that result in indeterminate forms (0/0) are automatically resolved to a distance of 0.

Association Correction via Cramér's V: Standard categorical distances (like the matching coefficient) implicitly assume that all features provide independent pieces of information. When features are highly associated, this causes an overrepresentation of redundant information. Specifying method = "cramer" solves this problem by utilizing the inverse of the regularized Cramér's V association matrix as a Mahalanobis-type metric tensor.

Value

Depending on `output_format`, returns a `dist` object or a numeric matrix.

References

Grané A, Salini S, Infante G (2026). "New distances for mixed-type data able to cope with redundant information." *AStA Advances in Statistical Analysis*. doi:10.1007/s10182026005656.

See Also

[daisy](#) for underlying Gower computations.

Examples

```
# Load example dataset and drop the technical weight column 'w_loop'
data("Data_HC_contamination", package = "dbrubust")
df <- Data_HC_contamination[, 1:9]

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

# Load example dataset
data("Data_HC_contamination", package = "dbrubust")
df <- Data_HC_contamination[1:20, 1:9]

# Example 1: Euclidean distance
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
d_gower <- calculate_distances(
  df,
  method = "gower",
  output_format = "dist"
)
```

```
# Example 5: Minkowski distance with q = 3
d_minkowski <- calculate_distances(
  numeric_data,
  method = "minkowski",
  q = 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: Gower distance specifying asymmetric binary columns and variable weights
d_gower_custom <- calculate_distances(
  df,
  method = "gower",
  output_format = "dist",
  type = list(asymm = c(8, 9)),
  weights = c(1, 1, 1, 1, 2, 2, 2, 1, 1)
)

# Example 9: Weighted Cramér's V distance for categorical data
d_cramer <- calculate_distances(
  categorical_data,
  method = "cramer",
  output_format = "matrix"
)

# Example 10: Cramér's V distance with specific observation weights
obs_weights <- rep(1, nrow(categorical_data))
d_cramer_weighted <- calculate_distances(
  categorical_data,
  method = "cramer",
  weights = obs_weights,
  output_format = "dist"
)
```

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. doi:10.1016/j.seps.2024.101992.

Data_HC_no_contamination
<i>High-correlation dataset without contamination</i>

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. doi:10.1016/j.seps.2024.101992.

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. doi:10.1016/j.seps.2024.101992.

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. doi:10.1016/j.seps.2024.101992.

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 using the Lingoes correction.

Usage

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

Arguments

<code>D</code>	Numeric square matrix ($n \times n$) of pairwise squared distances. Must be symmetric with zeros on the diagonal.
<code>w</code>	Numeric vector of weights (length n). Internally normalized to sum to 1. If NULL (default), uniform weights are used ($w_i = 1/n$).
<code>tol</code>	Numeric tolerance for detecting negative eigenvalues (default: 1e-10).

Details

The function constructs the weighted Gram matrix $G_w = -0.5J_wDJ_w^T$, where J_w is the weighted centering matrix. If G_w has negative eigenvalues, D is not Euclidean. The correction consists of adding a constant $2 \cdot |\lambda_{\min}|$ to all off-diagonal entries of D .

This implementation is optimized for large datasets:

- It uses an $O(n^2)$ double-centering procedure that avoids explicitly creating the $n \times n$ centering matrix.
- It utilizes [eigs_sym](#) to compute only the smallest eigenvalue, avoiding the $O(n^3)$ cost of a full decomposition.
- A safety fallback to [eigen](#) is provided via tryCatch for numerical stability in edge cases.

Value

A list with components:

D_euc The corrected squared distance matrix.

min_eig The smallest eigenvalue of the Gram matrix before correction.

transformed Logical; TRUE if the Lingoes correction was applied.

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 classical multidimensional scaling." *Communications in Statistics-Theory and Methods*, **7**(13), 1233–1241.

See Also

[dist](#), [eigs_sym](#), [cmdscale](#)

Examples

```
# 1. Create a non-Euclidean distance matrix
set.seed(123)
n <- 50
points <- matrix(runif(n * 2), n, 2)
dist_mat <- as.matrix(dist(points))^2

# Introduce a violation of the triangle inequality
dist_mat[1, 2] <- dist_mat[2, 1] <- dist_mat[1, 2] * 10

# 2. Apply the Lingoes correction without weights (defaults to rep(1, n))
res <- make_euclidean(dist_mat)

# 4. Check results
if (res$transformed) {
  message("Correction applied. Smallest eigenvalue was: ", res$min_eig)
}
```

merge_distances

Multi-Source Distance Combination and Fusion via Global RelMS

Description

Combines multiple distance matrices (sources) into a single consensus Euclidean distance matrix. Depending on the RelMS parameter, the function either applies Related Metric Scaling (RelMS) to penalize cross-source redundancy or performs a direct weighted aggregation (Gower-like approach). Inputs and outputs are enforced to be Euclidean via [make_euclidean](#).

Usage

```
merge_distances(
  distance_list,
  matrix_weights = NULL,
  w = NULL,
  squared = FALSE,
  RelMS = TRUE,
  tol = 1e-10
)
```

Arguments

distance_list A list of matrix or dist objects. All matrices must be square (n x n) and Euclidean.

matrix_weights	A numeric vector of weights for each distance matrix in distance_list. Must be strictly positive and sum to 1. If NULL, defaults to equal weighting (1 / m).
w	A numeric vector of observation/sample weights (length n). Must be strictly positive and will be normalized to sum to 1. If NULL, uniform weights (1 / n) are used.
squared	Logical. If FALSE (default), inputs are standard distances and are squared internally during processing. If TRUE, inputs are treated as pre-squared distance matrices.
RelMS	Logical. If TRUE (default), applies RelMS cross-metric redundancy penalization using the C++ kernel. If FALSE, computes a weighted direct linear combination.
tol	Numeric tolerance passed to make_euclidean for eigenvalue decomposition (default: 1e-10).

Details

When integrating data from multiple domains (e.g., clinical measurements, genomic markers, and survey responses), individual features often share latent signals. Direct distance aggregation overweights redundant dimensions. Global RelMS addresses this by subtracting the pairwise cross-interactions of standardized Gram roots. The matrix operations required for cross-interaction terms are offloaded to C++ via RcppArmadillo for maximum performance on high-dimensional data.

Value

A square numeric matrix (n x n) containing the fused consensus distances (or squared distances if squared = TRUE).

References

Boj E, Grané A (2024). “The robustification of distance-based linear models: Some proposals.” *Socio-Economic Planning Sciences*, **95**, 101992. doi:10.1016/j.seps.2024.101992.

See Also

[make_euclidean](#)

Examples

```
# =====
# SCENARIO 1: GEOMETRIC EXACTNESS (3 POINTS ALONG A STRAIGHT LINE)
# =====
D1 <- matrix(c(0, 1, 5,
               1, 0, 4,
               5, 4, 0), nrow = 3, byrow = TRUE)
D2 <- 4 * D1 # Source 2 is 100% redundant (scaled x4)

# Weighted direct fusion (weights = 0.5, 0.5) with standard distances
merge_distances(list(D1, D2), matrix_weights = c(0.5, 0.5),
                RelMS = FALSE, squared = FALSE)
```

```

# Direct fusion on pre-squared matrices (squared = TRUE)
merge_distances(list(D1, D2), matrix_weights = c(0.5, 0.5),
                RelMS = FALSE, squared = TRUE)

## Not run:
# =====
# SCENARIO 2: MULTI-DOMAIN INTEGRATION (CLINICAL, GENOMIC, & SURVEY)
# =====
set.seed(2026)
n <- 100

# Simulated multi-domain datasets
df_clinical <- data.frame(age = rnorm(n, 55, 10), bp = rnorm(n, 120, 15))
df_genomic  <- as.data.frame(matrix(rbinom(n * 30, 1, 0.2), nrow = n))
df_survey   <- data.frame(diet = factor(sample(c("Low", "Med", "High"),
                                              n, replace = TRUE)))

# Compute specific distance matrices per data type
D_clinical <- calculate_distances(df_clinical, method = "euclidean",
                                output_format = "matrix")
D_genomic  <- calculate_distances(df_genomic, method = "manhattan",
                                output_format = "matrix")
D_survey   <- calculate_distances(df_survey,
                                method = "matching_coefficient",
                                output_format = "matrix")

dist_list_multi <- list(Clinical = D_clinical,
                       Genomic  = D_genomic,
                       Survey   = D_survey)

# Fusion with custom source weights and RelMS penalization
D_fused_multi <- merge_distances(
  distance_list = dist_list_multi,
  matrix_weights = c(0.5, 0.3, 0.2),
  RelMS         = TRUE
)
dim(D_fused_multi)

# =====
# SCENARIO 3: HIGH-DIMENSIONAL INTEGRATION OF 40 REDUNDANT SOURCES
# =====
n_samples <- 80
n_sources <- 40

# Shared unobserved latent factors present across all sources
latent_factors <- matrix(rnorm(n_samples * 2), nrow = n_samples, ncol = 2)

dist_40_sources <- vector("list", n_sources)
for (s in 1:n_sources) {
  signal <- latent_factors %*% rnorm(2)
  noise  <- matrix(rnorm(n_samples * 4, sd = 0.5),
                  nrow = n_samples, ncol = 4)
  dist_40_sources[[s]] <- as.matrix(dist(cbind(signal, noise)))
}

```

```

}

# Compare RelMS (redundancy penalization) vs Direct Linear Fusion
D_relms <- merge_distances(dist_40_sources, RelMS = TRUE)
D_direct <- merge_distances(dist_40_sources, RelMS = FALSE)

cat("Mean distance (RelMS = TRUE) :", mean(D_relms), "\n")
cat("Mean distance (RelMS = FALSE):", mean(D_direct), "\n")

## End(Not run)

```

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 using either Geometric Variability (GV) trimming or a Weighted Minimum Covariance Determinant (MCD) approach. 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"),
  cov_method = c("gv", "mcd"),
  robust_cov = NULL,
  alpha = 0.1,
  niter = 40,
  nstart = 20,
  return_dist = FALSE
)

```

Arguments

<code>data</code>	Data frame or numeric matrix containing the observations.
<code>cont_vars</code>	Character vector of column names for continuous variables.
<code>bin_vars</code>	Character vector of column names for binary variables.
<code>cat_vars</code>	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.
cov_method	Character string specifying the robust covariance method for continuous variables: either "gv" (Geometric Variability) or "mcd" (Minimum Covariance Determinant). Default is "gv".
robust_cov	Optional. Precomputed robust covariance matrix for continuous variables. If NULL, it will be estimated internally using the specified cov_method.
alpha	Numeric trimming proportion for robust covariance of continuous variables.
niter	Integer specifying the number of concentration steps (C-steps) per start. Only used if cov_method = "mcd". Default is 40.
nstart	Integer specifying the number of random initializations. Only used if cov_method = "mcd". Default is 20.
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.

References

Boj E, Grané A (2024). "The robustification of distance-based linear models: Some proposals." *Socio-Economic Planning Sciences*, **95**, 101992. doi:10.1016/j.seps.2024.101992.

Boj E, Grané A, Mayo-Iscar A (2026). "Robust distance-based generalized linear models: a new tool for classification." *Statistical Methods and Applications*. doi:10.1007/s10260026008601.

Examples

```
# Example: Robust Squared Distances for Mixed Data using different Covariance Methods

# Load example data and subset
data("Data_HC_contamination", package = "dbrubust")
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: Generalized Gower (ggower)
# -----
```

```

# 1A. Using Geometric Variability (gv) for continuous variables
dist_sq_ggower_gv <- robust_distances(
  data = Data_small,
  cont_vars = cont_vars,
  bin_vars = bin_vars,
  cat_vars = cat_vars,
  w = w,
  alpha = 0.10,
  method = "ggower",
  cov_method = "gv"
)

# 1B. Using Minimum Covariance Determinant (mcd) - Set seed for reproducibility
set.seed(123)
dist_sq_ggower_mcd <- robust_distances(
  data = Data_small,
  cont_vars = cont_vars,
  bin_vars = bin_vars,
  cat_vars = cat_vars,
  w = w,
  alpha = 0.10,
  method = "ggower",
  cov_method = "mcd",
  niter = 40,
  nstart = 20
)

# Compare first 3x3 block of original GGower distances (GV vs MCD)
cat("GGower-GV squared distances (3x3 block):\n")
print(round(dist_sq_ggower_gv[1:3, 1:3], 4))
cat("\nGGower-MCD squared distances (3x3 block):\n")
print(round(dist_sq_ggower_mcd[1:3, 1:3], 4))

# -----
# Method 2: Related Metric Scaling (relms)
# -----

# 2A. Using Geometric Variability (gv) for continuous variables
dist_sq_relms_gv <- robust_distances(
  data = Data_small,
  cont_vars = cont_vars,
  bin_vars = bin_vars,
  cat_vars = cat_vars,
  w = w,
  alpha = 0.10,
  method = "relms",
  cov_method = "gv"
)

# 2B. Using Minimum Covariance Determinant (mcd) - Set seed for reproducibility
set.seed(123)
dist_sq_relms_mcd <- robust_distances(
  data = Data_small,

```

```

    cont_vars = cont_vars,
    bin_vars = bin_vars,
    cat_vars = cat_vars,
    w = w,
    alpha = 0.10,
    method = "relms",
    cov_method = "mcd",
    niter = 40,
    nstart = 20
  )

# Apply Euclidean correction if needed (e.g., on RelMS-MCD distances)
res_relms_mcd <- make_euclidean(dist_sq_relms_mcd, w)

cat("\nRelMS-MCD corrected squared distances (3x3 block):\n")
print(round(res_relms_mcd$D_euc[1:3, 1:3], 4))

```

visualize_distances	<i>Visualize Distance Matrices via MDS, Heatmap, or Network Graph</i>
---------------------	---

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,
  group_colors = 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 $\leq \min(4, n_{\text{obs}} - 1)$.
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.
group_colors	Optional character vector of custom colors for the groups. If NULL, a predefined custom palette is used.
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.

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)
# -----
custom_cols <- c("setosa" = "#1B9E77", "versicolor" = "#D95F02", "virginica" = "#7570B3")

visualize_distances(
  dist_mat = dist_iris,
  method = "mds_classic",
  k = 2,
  group = group_species,
  group_colors = custom_cols,,
  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

* datasets

- Data_HC_contamination, [5](#)
- Data_HC_no_contamination, [7](#)
- Data_MC_contamination, [8](#)
- Data_MC_no_contamination, [9](#)

calculate_distances, [2](#)

cmdscale, [11](#), [18](#)

daisy, [4](#)

Data_HC_contamination, [5](#)

Data_HC_no_contamination, [7](#)

Data_MC_contamination, [8](#)

Data_MC_no_contamination, [9](#)

dist, [11](#)

eigen, [10](#)

eigs_sym, [10](#), [11](#)

ggpairs, [18](#)

make_euclidean, [10](#), [11](#), [12](#), [14](#)

merge_distances, [11](#)

pheatmap, [18](#)

qgraph, [18](#)

robust_distances, [14](#)

visualize_distances, [17](#)

wcmdscale, [18](#)